AFRL-IF-RS-TR-2006-264
Final Technical Report
August 2006

NEW ARCHITECTURES, ALGORITHMS AND DESIGNS THAT LEAD TO IMPLEMENTED MACHINE REASONING OVER KNOWLEDGE IN EPISTEMIC AND DEONTIC FORMATS, IN THE SERVICE OF ADVANCED WARGAMING

Rensselaer Polytechnic Institute

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2006-264 has been reviewed and is approved for publication.

APPROVED: /s/

DAVID O. ROSS
Project Engineer

FOR THE DIRECTOR: /s/

JAMES W. CUSACK
Chief, Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
AUG 2006	Final	Aug 04 – Sep 05

4. TITLE AND SUBTITLE
NEW ARCHITECTURES, ALGORITHMS AND DESIGNS THAT LEAD TO IMPLEMENTED MACHINE REASONING OVER KNOWLEDGE IN EPISTEMIC AND DEONTIC FORMATS, IN THE SERVICE OF ADVANCED WARGAMING

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8750-04-2-0165

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)
Selmer Bringsjord, Konstantine Arkoudas and Yingrui Yang

5d. PROJECT NUMBER
558B

5e. TASK NUMBER
3G

5f. WORK UNIT NUMBER
WG

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rensselaer Polytechnic Institute (RPI)
110 Eighth Street
Troy NY 12180

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/IFSB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2006-264

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 06-577

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This effort investigated new architectures, algorithms, and designs to lead to: implemented machine reasoning over knowledge in expressive formats that include doxastic and deontic operators; understanding use of such implementations in a multi-agent setting; and implementation into other relevant DoD-related efforts. The RASCALS cognitive architecture has been developed further; new algorithms have been devised in mechanized epistemic and deontic logics; and the basic design for a logicist intelligent agent, callable from other systems, has been implemented.

15. SUBJECT TERMS
Algorithms, machine reasoning, doxastic and deontic operators, mechanized epistemic and deontic logics, logicist intelligent agent

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			David O. Ross
U	U	U	UL	48	19b. TELEPONE NUMBER (*Include area code*)

Table of Contents

Part I
What is in This Report?

This document, combined with associated code (sent electronically earlier under separate cover, and — in the case of Athena — acquired online), constitutes the final report for:

> **Project name**: "New Architectures, Algorithms, and Designs that Lead to Implemented Machine Reasoning over Knowledge in Epistemic & Deontic Formats, in the Service of Advanced Wargaming"
>
> **PIs**: Selmer Bringsjord, Konstantine Arkoudas, Yingrui Yang. (POC: Bringsjord.)
>
> **Grant total**: $75,000 (includes overhead)

The structure of this report is as follow:

- Summary of Work Performed, Indexed to SOW.
- Paper #1, on multi-agent reasoning in connection with epistemic logic: "Metareasoning for Multi-Agent Epistemic Logics."[1]

  Hereafter referred to as simply 'LNAI paper.'
- Paper #2, on an advanced synthetic character (E) in connection with the RASCALS cognitive architecture: "Advanced Synthetic Characters, Evil, and E."[2]

  Hereafter referred to as simply 'GameOn-2005 paper.'
- Paper #3, on mechanizing deontic logic: "Toward Ethical Robots via Mechanized Deontic Logic"

  Hereafter referred to as simply 'AAAI-FS paper.'
- Paper #4, on generalizing our mechanizing of deontic logic: , about to appear in *IEEE Intelligent Systems*.[3]

  Hereafter referred to as simply 'IEEE paper.'
- Athena tutorial (in-person demos of Athena at AFRL to follow in May 2006, tentatively scheduled specifically for the week of May 8)
- Explanation of associated code (in-person demos at AFRL took place in May 2006). Materials (Windows executable of l-agent derived from Slate, tutorial transcript of simple session recorded demo of the executable as a movie, and two `ath` files) are available at:

  http://www.cogsci.rpi.edu/research/rair/wargaming/

[1] The full reference is: Arkoudas, A. & Bringsjord. S. (2005) "Metareasoning for Multi-agent Epistemic Logics," in *Lecture Notes in Artificial Intelligence (LNAI)*, (New York, NY: Springer–Verlag), pp. 111–125.

[2] The full reference is: Bringsjord, S., Khemlani, S., Arkoudas, K., McEvoy, C., Destefano, M., Daigle, M. (2005) "Advanced Synthetic Characters, Evil, and E," *Game-On 2005, 6^{th} International Conference on Intelligent Games and Simulation*, (Ghent-Zwijnaarde, Belgium: European Simulation Society), pp. 31–39.

[3] The paper is in production at present.

Part II
Work Performed, Indexed to SOW

1 Overall Objective

The overall objective of this project, pulled verbatim from the SOW:

> 1.1 The objective of this effort is to investigate new architectures, algorithms, and designs to lead to: implemented machine reasoning over knowledge in expressive formats that include doxastic (= epistemic) and deontic operators; understanding of the tractability of using such implementations in a multi-agent setting; and transfer of such architectures, algorithms, and eventual implementations into other relevant DoD-related efforts.

Numerous such investigations have been conducted. The RASCALS cognitive architecture has been developed further, in connection with the synthetic character known simply as 'E'; new algorithms have been devised in the area of mechanized epistemic and deontic logics; and the basic design for a logicist intelligent agent (or just an l-agent), callable from other systems, has been implemented. Corresponding to these achievements, which are associated with the more specific points in the SOW (as explained below), are published papers and code made available to AFRL. The papers are included in this document, and the code for l-agents, contextualized in this document, is provided at

> http://www.cogsci.rpi.edu/research/rair/wargaming/

At this location, a Windows executable is provided, as well as a transcript of a simple session, a recorded tutorial demonstration, and two `ath` files.

2 Technical Requirements from the SOW

2.1 Requirement 4.1

> 4.1 The contractor shall design, develop, document, demonstrate, and deliver a machine reasoning capability to help support construction of cognitively robust intelligent agents; specifically, this capability will allow agents to reason over doxastic and deontic information. This capability will be realized in the contractor's pre-existing tools and systems ("contractors systems"). These pre-existing systems are owned by the contractor, and include: Athena, MARMML, Slate, and RASCALS. Other pre-existing systems will be used as well, and are in the public domain (e.g., SNARK).

The capability in question is detailed in two papers included here (LNAI and AAAI-FS/IEEE). Reasoning over doxastic information is detailed in the LNAI paper. Reasoning over deontic information is detailed in the AAAI-FS/IEEE paper. Corresponding `ath` files are provided. This brings us to the more specific sub-requirement here:

2.1.1 Requirement 4.1.1

> *Rendering Scenarios Expressed Doxastic Systems (e.g., KD45) in Computational Form via Logic-Based AI Techniques.* The contractor shall develop the theoretical constructions (architectures, algorithms, designs, etc.) necessary for the computational implementation, in the contractor's systems, of test scenarios expressed in the modal logic KD45 (and/or other such logics) of belief and knowledge. This logic allows for reasoning over doxastic information, which is information about what agents believe and know. The contractor shall provide a method to address the technical problem known as "logical omniscience."

The theoretical constructions called for in 4.1.1 have been invented, and are presented in the LNAI paper. The implementation has been accomplished as well; it is reported in the LNAI paper, and, again, the associated code is provided on the RAIR Lab's web site (see Part VIII). The next sub-requirement under 4.1 is:

2.1.2 Requirement 4.1.2

> *Rendering Scenarios Expressed Deontic Systems (e.g., DSDL3) in Computational Form via Logic-Based AI Techniques.* The contractor shall develop the theoretical constructions necessary for the computational implementation, in the contractors systems, of test scenarios expressed in the modal logic DSDL3 (Lewis 1974) (and/or other such logics) of obligation. This logic allows for reasoning over deontic information, which is information about the moral status of actions. The contractor shall address the technical problem known as "adequacy" (or conditional obligation).

The theoretical constructions alluded to here have been invented, and are specified in the AAAI/IEEE paper. In addition, the implementation has been accomplished. The relevant Athena code is provided on the RAIR Lab web site (see Part VIII).

2.1.3 Requirement 4.1.3

> *Long-Term Use by AFRL.* The contractor will investigate the design of an application programming interface (API) that allows developers working in the Java programming language at AFRL to exploit the capability referred to in section 4.1. Such an API will make possible the long-term use of this capability in conjunction with the contractors systems at AFRL. Members of AFRL's Third Generation Wargaming Group (3GWG) will advise the contractor about the conditions the API must satisfy, and will make available its Java-based systems, and specifications thereof, to the contractor, in order to enable the design of API by the contractor.

Conditions that the API would need to satisfy have not been provided, and the Java-based systems, and specifications thereof, have not been provided by AFRL's 3GWG. (Are the systems in question still under development at AFRL Rome?) However, the RAIR Lab has nonetheless designed and built a number of functions that would be central parts of any calls out from 3GWG Java-based systems to our reasoning technology. These functions are provided on the RAIR Lab's web site, and can be demonstrated and explained in person in the planned upcoming visit on or shortly after May 8. Calls out from any Java-based system should follow the interoperability standards currently in place in DTO, which are based on the ISO Common Logic standard, and are gaining steam as an across-the-board standard for DoD R&D. Readers are once again referred to Part VIII.

2.1.4 Requirement 4.1.4

> *Testbed Development.* The contractor shall document the appropriate set of test scenarios referred to in sections 2.1.1 and 2.1.3, coordinated with the Government, for the reasoning technology developed. Some of these test scenarios will involve doxastic information; some will involve deontic information.
>
> - 4.1.4.1 The contractor shall deliver all applications developed and associated software implementations of the modal logics described in the form of computer code executable in the contractor's systems.
> - 4.1.4.2 The contractor shall develop and deliver a detailed technical tutorial and maintenance document for the contractors systems.

The test scenarios are detailed in the LNAI (doxastic) and AAAI/IEEE (deontic) papers. Once again, code is available on the RAIR Lab's web site. Regarding tutorial information, a full Athena tutorial is included in this document. In-person tutorials will be provided as well (tentatively planned for the week of May 8 2006).

Metareasoning for multi-agent epistemic logics

Konstantine Arkoudas and Selmer Bringsjord

RPI
{arkouk,brings}@rpi.edu

Abstract. We present an encoding of a sequent calculus for a multi-agent epistemic logic in Athena, an interactive theorem proving system for many-sorted first-order logic. We then use Athena as a metalanguage in order to reason about the multi-agent logic an as object language. This facilitates theorem proving in the multi-agent logic in several ways. First, it lets us marshal the highly efficient theorem provers for classical first-order logic that are integrated with Athena for the purpose of doing proofs in the multi-agent logic. Second, unlike model-theoretic embeddings of modal logics into classical first-order logic, our proofs are directly convertible into native epistemic logic proofs. Third, because we are able to quantify over propositions and agents, we get much of the generality and power of higher-order logic even though we are in a first-order setting. Finally, we are able to use Athena's versatile tactics for proof automation in the multi-agent logic. We illustrate by developing a tactic for solving the generalized version of the wise men problem.

1 Introduction

Multi-agent modal logics are widely used in Computer Science and AI. Multi-agent epistemic logics, in particular, have found applications in fields ranging from AI domains such as robotics, planning, and motivation analysis in natural language [13]; to negotiation and game theory in economics; to distributed systems analysis and protocol authentication in computer security [16, 31]. The reason is simple—intelligent agents must be able to reason about knowledge. It is therefore important to have efficient means for performing machine reasoning in such logics. While the validity problems for most propositional modal logics are of intractable theoretical complexity[1], several approaches have been investigated in recent years that have resulted in systems that appear to work well in practice. These approaches include tableau-based provers, SAT-based algorithms, and translations to first-order logic coupled with the use of resolution-based theorem provers. Some representative systems are Fact [24], KSATC [14], TA [25], LWB [23], and MSPASS [38].

Tranlation-based approaches (such as that of MSPASS) have the advantage of leveraging the tremendous implementation progress that has occurred over

[1] For instance, the validity problem for multi-agent propositional epistemic logic is PSPACE-complete [18]; adding a common knowledge operator makes the problem EXPTIME-complete [21].

the last few decades in first-order theorem proving. Soundness and completeness are ensured by the soundness and completeness of the resolution prover (once the soundness and completeness of the translation have been shown), while a decision procedure is automatically obtained for any modal logic that can be translated into a decidable fragment of first-order logic (such as the two-variable fragment). Furthermore, Kripke frames are first-order definable [17], so translating from a modal setting to the classical first-order setting is fairly straightforward. For instance, the well-known formula $[\Box P \wedge \Box(P \Rightarrow Q)] \Rightarrow \Box Q$ becomes

$$\begin{aligned}\forall\ w_1\ .\ [&(\forall\ w_2\ .\ R(w_1, w_2) \Rightarrow P(w_2)) \wedge \\ &(\forall\ w_2\ .\ R(w_1, w_2) \Rightarrow P(w_2) \Rightarrow Q(w_2))] \Rightarrow (\forall\ w_2\ .\ R(w_1, w_2) \Rightarrow Q(w_2))\end{aligned}$$

Here the variables w_1 and w_2 range over possible worlds, and the relation R represents Kripke's accessibility relation. A constant propositional atom P in the modal language becomes a unary predicate $P(w)$ that holds (or not) for a given world w.

This is the classical translation of modal logic into first-order logic [18], and we might say that it is a *semantic* embedding, since the Kripke semantics of the modal language are explicitly encoded in the translated result. This is, for instance, the approach taken by McCarthy in his "Formalizing two puzzles involving knowledge" [30]. A drawback of this approach is that proofs produced in the translated setting are difficult to convert back into a form that makes sense for the user in the original modal setting, altough alternative translation techniques such as the functional translation to path logic can alleviate this issue in some cases [39]. Another drawback is that if a result is not obtained within a reasonable amount of time (which is almost certain to happen when no decision procedure is available, as in first-order modal logics), then a batch-oriented prover is of little help to the user due to its "low bandwidth of interaction" [12]. Much greater flexibilitInteractive proof systems such as PVS [37], HOL [20], Isabelle [34], and Athena [2] that offer tactics, facilities for goal decomposition and computation,

In this paper we explore another approach: We embed a multi-agent epistemic logic into many-sorted first-order logic in a proof-theoretic rather than in a model-theoretic way. [2] Specifically, we use the interactive theorem proving system Athena (which is briefly reviewed in the Appendix) to encode the formulas of the epistemic logic along with the inference rules of a sequent calculus for it. Hence first-order logic becomes our metalanguage and the epistemic logic becomes our object language. We then use standard first-order reasoning (our metalanguage) to reason about proofs in the object logic. In effect, we end up reasoning about reasoning—hence the term *metareasoning*. Since our metareasoning occurs at the standard first-order level, we are free to leverage existing theorem-proving systems for automated deduction. In particular, we make heavy

[2] This paper treats a propositional logic of knowledge, but the technique can be readily applied to full first-order multi-agent epistemic logic, and indeed to hybrid multimodal logics, e.g., combination logics for temporal and epistemic reasoning.

use of Vampire [41], a cutting-edge resolution-based prover that is seamlessly integrated with Athena.

Our approach has two additional advantages. First, it is trivial to translate the constructed proofs into modal form, since the Athena proofs are already *about proofs in the modal logic*. Second, because the abstract syntax of the epistemic logic is explicitly encoded in Athena, we can quantify over propositions, sequents, and agents. Accordingly, we get the generalization benefits of higher-order logic even in a first-order setting. This can result in significant efficiency improvements. For instance, in solving the generalized wise men puzzle it is necessary at some point to derive the conclusion $M_2 \vee \cdots \vee M_n$ from the three premises $\neg K_\alpha(M_1)$, $K_\alpha(\neg(M_2 \vee \cdots \vee M_n) \Rightarrow M_1)$, and

$$\neg(M_2 \vee \cdots \vee M_n) \Rightarrow K_\alpha(\neg(M_2 \vee \cdots \vee M_n))$$

where $M_1, \ldots, M_n$ are atomic propositions and α is an epistemic agent, $n > 1$. In the absence of an explicit embedding of the epistemic logic, this would have to be done with a tactic that accepted a list of propositions $[M_1 \cdots M_n]$ as input and performed the appropriate deduction dynamically, which would require an amount of effort quadratic in the length of the list. By contrast, in our approach we are able to formulate and prove a "higher-order" lemma stating

$$\forall\, P, Q, \alpha\ .\ \{\neg K_\alpha(P), K_\alpha(\neg Q \Rightarrow P), \neg Q \Rightarrow K_\alpha(\neg Q)\} \vdash Q$$

Obtaining the desired conclusion for any given $M_1, \ldots, M_n$ then becomes a matter of instantiating this lemma with $P \mapsto M_1$ and $Q \mapsto M_2 \vee \cdots \vee M_n$. We have thus reduced the asymptotic complexity of our task from quadratic time to constant time.

But perhaps the most distinguishing aspect of our work is our emphasis on *tactics*. Tactics are proof algorithms, which, unlike conventional algorithms, are guaranteed to produce sound results. That is, if and when a tactic outputs a result P that it claims to be a theorem, we can be assured that P is indeed a theorem. Tactics are widely used for proof automation in first- and higher-order proof systems such as HOL [20] and Isabelle [34]. In Athena tactics are called *methods*, and are particularly easy to formulate owing to Athena's Fitch-style natural deduction system and its assumption-base semantics [3]. A major goal of our research is to find out how easy—or difficult—it may be to automate multi-agent modal logic proofs with tactics. Our aim is not to obtain a completely automatic decision procedure for a certain logic (or class of logics), but rather to enable efficient interactive—i.e., semi-automatic—theorem proving in such logics for challenging problems that are beyond the scope of completely automatic provers. In this paper we formulate an Athena tactic for solving the generalized version of the wise men problem (for any given number of wise men). The relative ease with which this method was formulated is encouraging.

The remainder of this paper is structured as follows. In the next section we present a sequent calculus for the epistemic logic that we will be encoding. In Section 3 we present the wise men puzzle and formulate an algorithm for solving the generalized version of it in the sequent calculus of Section 2. In Section 4

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q}\ [\wedge\text{-}I] \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P}\ [\wedge\text{-}E_1] \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q}\ [\wedge\text{-}E_2]$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q}\ [\vee\text{-}I_1] \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q}\ [\vee\text{-}I_2]$$

$$\frac{\Gamma \vdash P_1 \vee P_2 \quad \Gamma, P_1 \vdash Q \quad \Gamma, P_2 \vdash Q}{\Gamma \vdash Q}\ [\vee\text{-}E]$$

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q}\ [\Rightarrow\text{-}I] \qquad \frac{\Gamma \vdash P \Rightarrow Q \quad \Gamma \vdash P}{\Gamma \vdash Q}\ [\Rightarrow\text{-}E]$$

$$\frac{\Gamma \vdash \neg\neg P}{\Gamma \vdash P}\ [\neg\text{-}E] \qquad \frac{\Gamma, P \vdash \bot}{\Gamma \vdash \neg P}\ [\neg\text{-}I] \qquad \frac{}{\Gamma, P \vdash P}\ [\textit{Reflex}]$$

$$\frac{\Gamma \vdash P}{\Gamma \cup \Gamma' \vdash P}\ [\textit{Dilution}] \qquad \frac{\Gamma \vdash P \wedge \neg P}{\Gamma \vdash \bot}\ [\bot\text{-}I] \qquad \frac{}{\Gamma \vdash \top}\ [\top\text{-}I]$$

Fig. 1. Inference rules for the propositional connectives.

we discuss the Athena encoding of the epistemic logic and present the Athena method for solving the generalized wise men problem. Finally, in Section 5 we consider related work.

2 A sequent formulation of a multi-agent epistemic logic

We will use the letters P, Q, $R, \ldots$, to designate arbitrary *propositions*, built according to the following abstract grammar:

$$P ::= A \mid \top \mid \bot \mid \neg P \mid P \wedge Q \mid P \vee Q \mid P \Rightarrow Q \mid K_\alpha(P) \mid C(P)$$

where A and α range over a countable set of atomic propositions ("atoms") and a primitive domain of *agents*, respectively. Propositions of the form $K_\alpha(P)$ and $C(P)$ are read as follows:

$K_\alpha(P)$: *agent* α *knows proposition* P

$C(P)$: *it is common knowledge that* P *holds*

By a *context* we will mean a finite set of propositions. We will use the letter Γ to denote contexts. We define a *sequent* as an ordered pair $\langle \Gamma, P \rangle$ consisting of a context Γ and a proposition P. A more suggestive notation for such a sequent

$$\frac{}{\Gamma \vdash [K_\alpha(P \Rightarrow Q)] \Rightarrow [K_\alpha(P) \Rightarrow K_\alpha(Q)]}\ [K] \qquad \frac{}{\Gamma \vdash K_\alpha(P) \Rightarrow P}\ [\mathrm{T}]$$

$$\frac{\emptyset \vdash P}{\Gamma \vdash C(P)}\ [C\text{-}I] \qquad \frac{}{\Gamma \vdash C(P) \Rightarrow K_\alpha(P)}\ [C\text{-}E]$$

$$\frac{}{\Gamma \vdash [C(P \Rightarrow Q)] \Rightarrow [C(P) \Rightarrow C(Q)]}\ [C_K] \qquad \frac{}{\Gamma \vdash C(P) \Rightarrow C(K_\alpha(P))}\ [R]$$

Fig. 2. Inference rules for the epistemic operators.

is $\Gamma \vdash P$. Intuitively, this is a judgment stating that P follows from Γ. We will write P, Γ (or Γ, P) as an abbreviation for $\Gamma \cup \{P\}$. The sequent calculus that we will use consists of a collection of inference rules for deriving judgments of the form $\Gamma \vdash P$. Figure 1 shows the inference rules that deal with the standard propositional connectives. This part is standard (e.g., it is very similar to the sequent calculus of Ebbinghaus et al. [15]). In addition, we have some rules pertaining to K_α and C, shown in Figure 2.

Rule $[K]$ is the sequent formulation of the well-known *Kripke axiom* stating that the knowledge operator distributes over conditionals. Rule $[C_K]$ is the corresponding principle for the common knowledge operator. Rule $[T]$ is the "truth axiom": an agent cannot know false propositions. Rule $[C_I]$ is an introduction rule for common knowledge: if a proposition P follows from the empty set of hypotheses, i.e., if it is a tautology, then it is commonly known. This is the common-knowledge version of the "omniscience axiom" for single-agent knowledge which says that $\Gamma \vdash K_\alpha(P)$ can be derived from $\emptyset \vdash P$. We do not need to postulate that axiom in our formulation, since it follows from $[C\text{-}I]$ and $[C\text{-}E]$. The latter says that if it is common knowledge that P then any (every) agent knows P, while $[R]$ says that if it is common knowledge that P then it is common knowledge that (any) agent α knows it. $[R]$ is a reiteration rule that allows us to capture the recursive behavior of C, which is usually expressed via the so-called "induction axiom"

$$C(P \Rightarrow E(P)) \Rightarrow [P \Rightarrow C(P)]$$

where E is the shared knowledge operator. Since we do not need E for our purposes, we omit its formalization and "unfold" C via rule $[R]$ instead.
We state a few lemmas that will come handy later:

Lemma 1 (Cut). *If* $\Gamma_1 \vdash P_1$ *and* $\Gamma_2, P_1 \vdash P_2$ *then* $\Gamma_1 \cup \Gamma_2 \vdash P_2$.

Proof: Assume $\Gamma_1 \vdash P_1$ and $\Gamma_2, P_1 \vdash P_2$. Then, by $[\Rightarrow\text{-}I]$, we get $\Gamma_2 \vdash P_1 \Rightarrow P_2$. Further, by dilution, we have $\Gamma_1 \cup \Gamma_2 \vdash P_1 \Rightarrow P_2$ and $\Gamma_1 \cup \Gamma_2 \vdash P_1$. Hence, by $[\Rightarrow\text{-}E]$, we obtain $\Gamma_1 \cup \Gamma_2 \vdash P_2$. □

The proofs of the remaining lemmas are equally simple exercises:

Lemma 2 (⇒-transitivity). *If* $\Gamma \vdash P_1 \Rightarrow P_2$ *and* $\Gamma \vdash P_2 \Rightarrow P_3$ *then* $\Gamma \vdash P_1 \Rightarrow P_3$.

Lemma 3 (contrapositive). *If* $\Gamma \vdash P \Rightarrow Q$ *then* $\Gamma \vdash \neg Q \Rightarrow \neg P$.

Lemma 4. *(a)* $\emptyset \vdash (P_1 \vee P_2) \Rightarrow (\neg P_2 \Rightarrow P_1)$*; and* (b) $\Gamma \vdash C(P_2)$ *whenever* $\emptyset \vdash P_1 \Rightarrow P_2$ *and* $\Gamma \vdash C(P_1)$.

Lemma 5. *For all* P, Q*, and* Γ*,* $\Gamma \vdash [C(P) \wedge C(Q)] \Rightarrow C(P \wedge Q)$.

3 The generalized wise men puzzle

Consider first the three-men version of the puzzle:

> Three wise men are told by their king that at least one of them has a white spot on his forehead. In reality, all three have white spots on their foreheads. We assume that each wise man can see the others' foreheads but not his own, and thus each knows whether the others have white spots. Suppose we are told that the first wise man says, "I do not know whether I have a white spot," and that the second wise man then says, "I also do not know whether I have a white spot." Now consider the following question: Does the third wise man now know whether or not he has a white spot? If so, what does he know, that he has one or doesn't have one?

This version is essentially identical to the muddy-children puzzle, the only difference being that the declarations of the wise men are made sequentially, whereas in the muddy-children puzzle, the children proclaim what they know (or not know) in parallel at every round.

In the generalized version of the puzzle we have an arbitrary number $n+1$ of wise men $w_1, \ldots, w_{n+1}$, $n \geq 1$. They are told by their king that at least one them has a white spot on his forehead. Again, in actuality they all do. And they can all see one another's foreheads, but not their own. Supposing that each of the first n wise men, $w_1, \ldots, w_n$, sequentially announces that he does not know whether or not he has a white spot on his forehead, the question is what would the last wise man w_{n+1} report.

For all $n \geq 1$, it turns out that the last—$(n+1)^{st}$—wise man knows he is marked. The case of two wise men is simple. The reasoning runs essentially by contradiction. The second wise man reasons as follows:

> Suppose I were not marked. Then w_1 would have seen this, and knowing that at least one of us is marked, he would have inferred that he was the marked one. But w_1 has expressed ignorance; therefore, I must be marked.

Consider now the case of $n = 3$ wise men w_1, w_2, w_3. After w_1 announces that he does not know that he is marked, w_2 and w_3 both infer that at least one of them is marked. For if neither w_2 nor w_3 were marked, w_1 would have seen this

and would have concluded—and stated—that he was the marked one, since he knows that at least one of the three is marked. At this point the puzzle reduces to the two-men case: both w_2 and w_3 know that at least one of them is marked, and then w_2 reports that he does not know whether he is marked. Hence w_3 proceeds to reason as previously that he is marked.

In general, consider $n+1$ wise men $w_1, \ldots, w_n, w_{n+1}, n \geq 1$. After the first j wise men $w_1, \ldots, w_j$ have announced that they do not know whether they are marked, for $j = 1, \ldots, n$, the remaining wise men $w_{j+1}, \ldots, w_{n+1}$ infer that at least one of them is marked. This holds for $j = n$ as well, which means that the last wise man w_{n+1} will infer (and announce, owing to his honesty) that he is marked.

The question is how to formalize this in our logic. Again consider the case of two wise men w_1 and w_2. Let $M_i, i \in \{1, 2\}$ denote the proposition that w_i is marked. For any proposition P, we will write $K_i(P)$ as an abbreviation for $K_{w_i}(P)$. We will only need three premises:

$$S_1 = C(\neg K_1(M_1))$$
$$S_2 = C(M_1 \vee M_2)$$
$$S_3 = C(\neg M_2 \Rightarrow K_1(\neg M_2))$$

The first premise says that it is common knowledge that the first wise man does not know whether he is marked. Although it sounds innocuous, note that a couple of assumptions are necessary to obtain this premise from the mere fact that w_1 has announced his ignorance. First, truthfulness—we must assume that the wise men do not lie, and further, that each one of them knows that they are all truthful. And second, each wise man must know that the other wise men will hear the announcement and believe it. Premise S_2 says that it is common knowledge that at least one of the wise men is marked. Observe that the announcement by the king is crucial for this premise to be justified. The two wise men can see each other and thus they individually know $M_1 \vee M_2$. However, each of them may not know that the other wise man knows that at least one of them is marked. For instance, w_1 may believe that he is not marked, and even though he sees that w_2 is marked, he may believe that w_2 does not know that at least one of them is marked, as w_2 cannot see himself. Finally, premise S_3 states that it is common knowledge that if w_2 is *not* marked, then w_1 will know it (because w_1 can see w_2). From these three premises we are to conclude that it is common knowledge that w_2 is marked. Symbolically, we need to derive the judgment $\{S_1, S_2, S_3\} \vdash C(M_2)$. If we have encoded the epistemic propositional logic in a predicate calculus, then we can achieve this immediately by instantiating Lemma 7 below with $\alpha \mapsto w_1$, $P \mapsto M_1$ and $Q \mapsto M_2$—without performing any inference whatsoever. This is what we have done in Athena.

For the case of $n = 3$ wise men our set of premises will be:

$$\begin{aligned}
S_1 &= C(\neg K_1(M_1)) \\
S_2 &= C(M_1 \vee M_2 \vee M_3) \\
S_3 &= C(\neg(M_2 \vee M_3) \Rightarrow K_1(\neg(M_2 \vee M_3))) \\
S_4 &= C(\neg K_2(M_2)) \\
S_5 &= C(\neg M_3 \Rightarrow K_2(\neg M_3))
\end{aligned}$$

Consider now the general case of $n + 1$ wise men $w_1, \ldots, w_n, w_{n+1}$. For any $i = 1, \ldots, n$, define

$$\begin{aligned}
S_1^i &= C(\neg K_i(M_i)) \\
S_2^i &= C(M_i \vee \cdots \vee M_{n+1}) \\
S_3^i &= C(\neg(M_{i+1} \vee \cdots \vee M_{n+1}) \Rightarrow K_i(\neg(M_{i+1} \vee \cdots \vee M_{n+1})))
\end{aligned}$$

The set of premises, which we will denote by Ω_{n+1}, can now be defined as

$$\Omega_{n+1} = \{C(M_1 \vee \cdots \vee M_{n+1})\} \bigcup_{i=1}^{n} \{S_1^i, S_3^i\}$$

Hence Ω_{n+1} has a total of $2n + 1$ elements. Note that S_2^1 is the commonly known disjunction $M_1 \vee \cdots \vee M_{n+1}$ and a known premise, i.e., a member of Ω_{n+1}. However, S_2^i for $i > 1$ is *not* a premise. Rather, it becomes derivable after the i^{th} wise man has made his announcement. Managing the derivation of these propositions and eliminating them via applications of the cut is the central function of the algorithm below. Before we present the algorithm we state a couple of key lemmas.

Lemma 6. *Consider any agent α and propositions P, Q, and let R_1, R_2, R_3 be the following three propositions:*

1. $R_1 = \neg K_\alpha(P)$;
2. $R_2 = K_\alpha(\neg Q \Rightarrow P)$;
3. $R_3 = \neg Q \Rightarrow K_\alpha(\neg Q)$

Then $\{R_1 \wedge R_2 \wedge R_3\} \vdash Q$.

Proof. By the following sequent derivation:

1.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash R_1$	[*Reflex*], $\wedge$-E_1
2.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash R_2$	[*Reflex*], $\wedge$-E_1, $\wedge$-E_2
3.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash R_3$	[*Reflex*], $\wedge$-E_2
4.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash K_\alpha(\neg Q) \Rightarrow K_\alpha(P)$	2, [K], $\Rightarrow$-E
5.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash \neg Q \Rightarrow K_\alpha(P)$	3, 4, Lemma 2
6.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash \neg K_\alpha(P) \Rightarrow \neg\neg Q$	5, Lemma 3
7.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash \neg\neg Q$	6, 1, $\Rightarrow$-E
8.	$\{R_1 \wedge R_2 \wedge R_3\} \vdash Q$	7, [$\neg$-E]

□

Lemma 7. *Consider any agent α and propositions P, Q. Define R_1 and R_3 as in Lemma 6, let $R_2 = P \vee Q$, and let $S_i = C(R_i)$ for $i = 1, 2, 3$. Then $\{S_1, S_2, S_3\} \vdash C(Q)$.*

Proof. Let $R_2' = \neg Q \Rightarrow P$ and consider the following derivation:

1.	$\{S_1, S_2, S_3\} \vdash S_1$	[*Reflex*]
2.	$\{S_1, S_2, S_3\} \vdash S_2$	[*Reflex*]
3.	$\{S_1, S_2, S_3\} \vdash S_3$	[*Reflex*]
4.	$\emptyset \vdash (P \vee Q) \Rightarrow (\neg Q \Rightarrow P)$	Lemma 4a
5.	$\{S_1, S_2, S_3\} \vdash C((P \vee Q) \Rightarrow (\neg Q \Rightarrow P))$	4, [C-I]
6.	$\{S_1, S_2, S_3\} \vdash C(P \vee Q) \Rightarrow C(\neg Q \Rightarrow P)$	5, [C_K], [$\Rightarrow$-E]
7.	$\{S_1, S_2, S_3\} \vdash C(\neg Q \Rightarrow P)$	6, 2, [$\Rightarrow$-E]
8.	$\{S_1, S_2, S_3\} \vdash C(\neg Q \Rightarrow P) \Rightarrow C(K_\alpha(\neg Q \Rightarrow P))$	[R]
9.	$\{S_1, S_2, S_3\} \vdash C(K_\alpha(\neg Q \Rightarrow P))$	8, 7, [$\Rightarrow$-E]
10.	$\{R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3\} \vdash Q$	Lemma 6
11.	$\emptyset \vdash (R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3) \Rightarrow Q$	10, [$\Rightarrow$-I]
12.	$\{S_1, S_2, S_3\} \vdash C((R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3) \Rightarrow Q)$	11, [C-I]
13.	$\{S_1, S_2, S_3\} \vdash C(R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3) \Rightarrow C(Q)$	12, [C_K], [$\Rightarrow$-E]
14.	$\{S_1, S_2, S_3\} \vdash C(R_1 \wedge K_\alpha(\neg Q \Rightarrow P) \wedge R_3)$	1, 3, 9, Lemma 5, [$\wedge$-I]
15.	$\{S_1, S_2, S_3\} \vdash C(Q)$	13, 14, [$\Rightarrow$-E]

□

Our method can now be stated as follows:

$\Phi \leftarrow \{S_1^1, S_2^1, S_3^1\}$;
$\Sigma \leftarrow \Phi \vdash S_2^2$;
Use Lemma 7 to derive Σ;
If $n = 1$ **halt**
else
  For $i = 2$ to n do
    begin
      $\Phi \leftarrow \Phi \cup \{S_1^i, S_3^i\}$;
      $\Sigma' \leftarrow \{S_1^i, S_2^i, S_3^i\} \vdash S_2^{i+1}$;
      Use Lemma 7 to derive Σ';
      $\Sigma'' \leftarrow \Phi \vdash S_2^{i+1}$;
      Use the cut on Σ and Σ' to derive Σ'';
      $\Sigma \leftarrow \Sigma''$
    end

The loop variable i ranges over the interval $2, \ldots, n$. For any i in that interval, we write Φ^i and Σ^i for the values of Φ and Σ upon conclusion of the i^{th} iteration of the loop. A straightforward induction on i will establish:

Lemma 8 (Algorithm correctness). *For any $i \in \{2, \ldots, n\}$,*

$$\Phi^i = \{C(M_1 \vee \cdots \vee M_{n+1})\} \bigcup_{j=1}^{i} \{S_1^j, S_3^j\}$$

while $\Sigma^i = \Phi^i \vdash S_2^{i+1}$.

It follows that $\Phi^n = \Omega_{n+1}$ and $\Sigma^n = \Phi^n \vdash S_2^{n+1} = \Omega_{n+1} \vdash S_2^{n+1}\Omega_{n+1} \vdash C(M_{n+1})$ which is our goal.

It is noteworthy that no such correctness argument is necessary in the formulation of the algorithm as an Athena method. Athena methods are guaranteed to be sound. As long as the produced result is of the right form (in our case, a sequent of the form $\Omega_{n+1} \vdash C(M_{n+1})$), we can be assured that the result follows logically from the contents of the assumption base.

4 Athena implementation

In this section we present the Athena encoding of the epistemic logic and our method for solving the generalized version of the wise men puzzle (refer to the Appendix for a brief review of Athena). We begin by introducing an uninterpreted domain of epistemic agents: `(domain Agent)`. Next we represent the abstract syntax of the propositions of the logic. The following Athena datatype mirrors the abstract grammar for propositions that was given in the beginning of Section 2:

```
(datatype Prop
  True
  False
  (Atom Boolean)
  (Not Prop)
  (And Prop Prop)
  (Or Prop Prop)
  (If Prop Prop)
  (Knows Agent Prop)
  (Common Prop))
```

We proceed to introduce a binary relation `sequent` that may obtain between a finite set of propositions and a single proposition:

```
(declare sequent (-> ((FSet-Of Prop) Prop) Boolean))
```

Here `FSet-Of` is a unary sort constructor: for any sort `T`, `(FSet-Of T)` is a new sort representing the set of all finite sets of elements of `T`. Finite sets are built with two polymorphic constructors: the constant `null`, representing the empty set; and the binary constructor `insert`, which takes an element x of sort `T` and a finite set S (of sort `(FSet-Of T)`) and returns the set $\{x\} \cup S$. We also have all the usual set-theoretic operations available (`union`, `intersection`, etc.).

The intended interpretation is that if `(sequent` S P`)` holds for a set of propositions S and a proposition P, then the sequent $S \vdash P$ is derivable in the epistemic logic via the rules presented in Section 2. Accordingly, we introduce axioms capturing those rules. For instance, the conjunction introduction rule is represented by the following axiom:

```
(define And-I
  (forall ?B ?P ?Q
    (if (and (sequent ?B ?P)
             (sequent ?B ?Q))
        (sequent ?B (And ?P ?Q)))))
```

Note that the lowercase `and` above is Athena's built-in conjunction operator, and hence represents conjunction at the metalanguage level, whereas `And` represents the object-level conjunction operator of the epistemic logic.

The cut rule and the common knowledge introduction (necessitation) rule become:

```
(define cut
  (forall ?B1 ?B2 ?P ?Q
    (if (and (sequent ?B1 ?P)
             (sequent (insert ?P ?B2) ?Q))
        (sequent (union ?B1 ?B2) ?Q))))

(define common-intro-axiom
  (forall ?P ?B
    (if (sequent null ?P)
        (sequent ?B (Common ?P)))))
```

The remaining rules are encoded by similar first-order axioms.

We next proceed to derive several lemmas that are useful for the proof. Some of these lemmas are derived completely automatically via the ATPs that are integrated with Athena. For instance, the cut rule is proved automatically (in about 10 seconds). As another example, the following result—part (b) of Lemma 4—is proved automatically:

```
(forall ?B ?P1 ?P2
  (if (and (sequent null (If ?P1 ?P2))
           (sequent ?B (Common ?P1)))
      (sequent ?B (Common ?P2))))
```

Other lemmas are established by giving natural deduction proofs. For instance, the proof of Lemma 6 in Section 3 is transcribed virtually verbatim in Athena, and validated in a fraction of a second. (The fact that the proof is abridged—i.e., multiple steps are compressed into single steps—is readily handled by invoking ATPs that automatically fill in any gaps.) Finally, we are able to prove Lemma 7, which is the key technical lemma. Utilizing the higher-order character of our encoding, we then define a method `main-lemma` that takes an arbitrary list of agents $[a_1 \cdots a_n], n \geq 1$, and specializes Lemma 7 with $P \mapsto M_{a_1}$, $Q \mapsto M_{a_2} \vee \cdots \vee M_{a_n}$, and $\alpha \mapsto a_1$ (recall that for any agent α, M_α signifies that α is marked). So, for instance, the application of `main-lemma` to the list $[a_1, a_2, a_3]$ would derive the conclusion $\{S_1, S_2, S_3\} \vdash C(M_{a_2} \vee M_{a_3})$, where $S_1 = C(\neg K_{a_1}(M_{a_1}))$, $S_2 = C(M_{a_1} \vee M_{a_2} \vee M_{a_3})$, and

$$S_3 = C(\neg(M_{a_2} \vee M_{a_3}) \Rightarrow K_{a_1}(\neg(M_{a_2} \vee M_{a_3})))$$

We also need a simple result `shuffle` asserting the equality $\Gamma, P_1, P_2 = \Gamma, P_2, P_1$ (i.e., $\Gamma \cup \{P_1\} \cup \{P_2\} = \Gamma \cup \{P_2\} \cup \{P_1\}$).

Using these building blocks, we express the tactic for solving the generalized wise men problem as the Athena method `solve` below. It takes as input a list of agents representing wise men, with at least two elements. Note that the for loop in the pseudocode algorithm has been replaced by recursion.

```
(define (solve wise-men)
  (dletrec
    ((loop (method (wise-men th)
             (dmatch wise-men
               ([_] (!claim th))
               ((list-of _ rest)
                 (dlet ((new-th (!main-lemma wise-men)))
                  (dmatch [th new-th]
                    ([(sequent context Q2)
                      (sequent (insert Q1
                                 (insert Q2 (insert Q3 null))) P)]
                     (dlet ((cut-th
                              (!derive (sequent
                                         (union
                                           context
                                           (insert Q1 (insert Q3 null)))
                                        P)
                                       [th new-th shuffle cut])))
                       (!loop rest cut-th)))))))))
     (dlet ((init (!prove-goal-2 wise-men)))
       (!loop (tail wise-men) init))))
```

Assuming that `w1`, `w2`, `w3` are agents representing wise men, invoking the method `solve` with the list `[w1 w2 w3])` as the argument will derive the appropriate result: $\Omega_3 \vdash$ `(Common (isMarked w3))`, where Ω_3 is the set of premises for the three-men case, as defined in the previous section.

5 Related work

The wise men problem became a staple of epistemic AI literature after being introduced by McCarthy [30]. Formalizations and solutions of the two-wise-men problem are found in a number of sources [26, 40, 19], most of them in simple multi-agent epistemic logics (without common knowledge). Several variations have been given; e.g., Konolige has a version in which the third wise man states that he does not know whether he is marked, but that he would know if only the second wise man were wiser [28]. Ballim and Wilks [8] solve the three-men version of the puzzle using the "nested viewpoints" framework. Vincenzo Pallotta's solution [33] is similar but his ViewGen framework facilitates agent simulation. Kim and Kowalski [27] use a Prolog-based implementation of metareasoning to solve the same version of the problem using common knowledge. A more natural proof was given by Aiello et al. [1] in a rewriting framework.

The importance of metareasoning and metaknowledge for intelligent agents is extensively discussed in "Logical foundations of Artifical Intelligence" by Genesereth and Nillson [19] (it is the subject of an entire chapter). They stress that the main advantage of an explicit encoding of the reasoning process is that it makes it possible to "create agents capable of reasoning in detail about the inferential abilities of and beliefs of other agents," as well as enabling introspection.

The only work we are aware of that has an explicit encoding of an epistemic logic in a rich metalanguage is a recent project [29] that uses the Calculus of Constructions (Coq [11]). However, there are important differences. First, they encode a Hilbert proof system, which has an adverse impact on the readability and writability of proofs. The second and most important difference is our emphasis on reasoning efficiency. The seamless integration of Athena with state-of-the-art provers such as Vampire and Spass is crucial for automation, as it enables the user to skip tedious steps and keep the reasoning at a high level of detail. Another distinguishing aspect of our work is our reliance on tactics. Athena uses a block-structured natural-deduction style not only for writing proofs but also for writing proof tactics ("methods"). Proof methods are much easier to write in this style, and play a key role in proof automation. Our emphasis on automation also differentiates our work from that of Basin et al. [9] using Isabelle, which only addresses proof presentation in modal logics, not automatic proof discovery.

References

1. L. C. Aiello, D. Nardi, and M. Schaerf. Yet another solution to the three wisemen puzzle. In *Proceedings of the 3rd International Symposium on Methodologies for Intelligent Systems*, pages 398–407, 1988.
2. K. Arkoudas. Athena. `http://www.cag.csail.mit.edu/~kostas/dpls/athena`.
3. K. Arkoudas. Denotational Proof Languages. PhD dissertation, MIT, 2000.
4. K. Arkoudas, S. Khurshid, D. Marinov, and M. Rinard. Integrating model checking and theorem proving for relational reasoning. In *Proceedings of the 7th International Seminar on Relational Methods in Computer Science (RelMiCS 7)*, Malente, Germany, May 2003.
5. K. Arkoudas and M. Rinard. Deductive runtime certification. In *Proceedings of the 2004 Workshop on Runtime Verification*, Barcelona, Spain, April 2004.
6. K. Arkoudas, K. Zee, V. Kuncak, and M. Rinard. On verifying a file system implementation. Technical Report CSAIL TR 946, MIT CSAIL, 2004.
7. T. Arvizo. A virtual machine for a type-ω denotational proof language. Masters thesis, MIT, June 2002.
8. A. Ballim and Y. Wilks. *Artificial Believers*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991.
9. David Basin, Seán Matthews, and Luca Viganò. A modular presentation of modal logics in a logical framework. In Jonathan Ginzburg, Zurab Khasidashvili, Carl Vogel, Jean-Jacques Lévy, and Enric Vallduví, editors, *The Tbilisi Symposium on Logic, Language and Computation: Selected Papers*, pages 293–307. CSLI Publications, Stanford, CA, 1998.
10. K. Claessen and N. Sorensson. New techniques that improve Mace-style finite model building. In *Model Computation—principles, algorithms, applications*, Miami, Florida, USA, 1973.
11. T. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76:95–120, 1988.
12. D. Cyrluk, S. Rajan, N. Shankar, , and M.K. Srivas. Effective theorem proving for hardware verification. In *Theorem Provers in Circuit Design (TPCD '94)*, volume 901 of *Lecture Notes in Computer Science*, pages 203–222, Bad Herrenalb, Germany, sep 1994. Springer-Verlag.

13. E. Davis and L. Morgenstern. Epistemic Logics and its Applications: Tutorial Notes. `www-formal.stanford.edu/leora/krcourse/ijcaitxt.ps`.
14. Giunchiglia E., Giunchiglia F., Sebastiani R., and Tacchella A. More evaluation of decision procedures for modal logics. In Cohn A. G., Schubert L., and Shapiro S. C., editors, *6th international conference on principles of knowledge representation and reasoning (KR'98)*, Trento, 2-5 June 1998.
15. H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, 2nd edition, 1994.
16. R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge*. MIT Press, Cambridge, Massachusetts, 1995.
17. M. Fitting. Basic modal logic. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Logical foundations*, volume 4 of *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford Science Publications, 1994.
18. D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. Many-dimensional modal logics: theory and applications. volume 4 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1994.
19. M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.
20. M. J. C. Gordon and T. F. Melham. *Introduction to HOL, a theorem proving environment for higher-order logic*. Cambridge University Press, Cambridge, England, 1993.
21. J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
22. M. Hao. Using a denotational proof language to verify dataflow analyses. Masters thesis, MIT, September 2002.
23. A. Heuerding. LWBtheory: information about some propositional logics via the WWW. *Logic Journal of the IGPL*, 4(4):169–174, 1996.
24. I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Sixth International Conference on Principles of Knowledge Representation and Reasoning*, pages 636–647, 1998.
25. U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In *Fifteenth International Joint Conference on Artificial Intelligence*, pages 202–209, 1997.
26. M. Huth and M. Ryan. *Logic in Computer Science: modelling and reasoning about systems*. Cambridge University Press, Cambridge, UK, 2000.
27. J. Kim and R. Kowalski. An application of amalgamated logic to multi-agent belief. In M. Bruynooghe, editor, *Second Workshop on Meta-Programming in Logic META90*, pages 272–283. 1990.
28. K. Konolige. *A deduction model of belief*. Research Notes in Artificial Intelligence. Pitman, London, UK, 1986.
29. Pierre Lescanne. Epistemic logic in higher order logic: an experiment with COQ. Technical Report RR2001-12, LIP-ENS de Lyon, 2001.
30. J. McCarthy. Formalization of two puzzles involving knowledge. In Vladimir Lifschitz, editor, *Formalizing Common Sense: Papers by John McCarthy*. Ablex Publishing Corporation, Norwood, New Jersey, 1990.
31. J.J. Meyer and W. Van Der Hoek. Epistemic Logic for Computer Science and Artificial Intelligence. volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.
32. D. Musser. Generic Software Design. `http://www.cs.rpi.edu/~musser/gsd`.
33. V. Pallotta. Computational dialogue Models. In *10th Conference of the European Chapter of the Association for Computational Linguistics EACL03*, 2003.

34. L. Paulson. *Isabelle, A Generic Theorem Prover*. Lecture Notes in Computer Science. Springer-Verlag, 1994.
35. F. J. Pelletier. A Brief History of Natural Deduction. *History and Philosophy of Logic*, 20:1–31, 1999.
36. M. Rinard and D. Marinov. Credible compilation with pointers. In *Proceedings of the 1999 Workshop on Run-Time Result Verification*, Trento, Italy, July 1999.
37. J. Rushby. PVS bibliography. Technical report, Computer Science Lab, SRI International, constantly updated at `http://www.csl.sri.com/pvs-bib.html`.
38. R. A. Schmidt. MSPASS. `http://www.cs.man.ac.uk/~schmidt/mspass/`, 1999.
39. R. A. Schmidt and U. Hustadt. Mechanised reasoning and model generation for extended modal logics. In H. C. M. de Swart, E. Orlowska, G. Schmidt, and M. Roubens, editors, *Theory and Applications of Relational Structures as Knowledge Instruments*, volume 2929 of *Lecture Notes in Computer Science*, pages 38–67. Springer, 2003.
40. D. Snyers and A. Thayse. Languages and logics. In A. Thayse, editor, *From modal logic to deductive databases*, pages 1–54. John Wiley & Sons, 1989.
41. A. Voronkov. The anatomy of Vampire: implementing bottom-up procedures with code trees. *Journal of Automated Reasoning*, 15(2), 1995.
42. C. Weidenbach. Combining superposition, sorts, and splitting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 2. North-Holland, 2001.

A Athena Overview

Athena is a new interactive theorem proving system that incorporates facilities for model generation, automated theorem proving, and structured proof representation and checking. It also provides a higher-order functional programming language, and a proof abstraction mechanism for expressing arbitrarily complicated inference *methods* in a way that guarantees soundness, akin to the tactics and tacticals of LCF-style systems such as HOL [20] and Isabelle [34]. Proof automation is achieved in two ways: first, through user-formulated proof methods; and second, through the seamless integration of state-of-the-art ATPs such as Vampire [41] and Spass [42] as primitive black boxes for general reasoning. For model generation, Athena integrates Paradox [10], a new highly efficient model finder. For proof representation and checking, Athena uses a block-structured Fitch-style natural deduction calculus [35] with novel syntactic constructs and a formal semantics based on the abstraction of *assumption bases* [3]. Most interestingly, a block-structured natural deduction format is used not only for writing proofs, but also for writing tactics (methods). This is a novel feature of Athena; all other tactic languages we are aware of are based on sequent calculi. Tactics in this style are considerably easier to write and remarkably useful in making proofs more modular and abstract.

Athena has been used to implement a proof-emitting optimizing compiler [36]; to integrate model checking and theorem proving for relational reasoning [4]; to implement various "certifying" algorithms [5]; to verify the core operations of a Unix-like file system [6]; to prove the correctness of dataflow analyses [22]; and to reason about generic software [32]. This section presents parts of Athena

relevant to understanding the code in this paper. A more thorough presentation of Athena's syntax and semantics can be found elsewhere [7].

In Athena, an arbitrary universe of discourse (sort) is introduced with a `domain` declaration, for example:

```
(domain Real)
(domain Person)
```

Function symbols and constants can then be declared on the domains, e.g.:

```
(declare + (-> (Real Real) Real))
(declare joe Person)
```

Relations are functions whose range is the predefined sort `Boolean`, e.g.,

```
(declare < (-> (Real Real) Boolean))
```

Inductively generated domains are introduced as *datatypes*, e.g.,

```
(datatype Nat
  zero
  (succ Nat))
```

Here `Nat` is freely generated by the *constructors* `zero` and `succ`. This is equivalent to issuing the declarations `(domain Nat)`, `(declare zero Nat)`,

```
(declare succ (-> (Nat) Nat))
```

and additionally postulating a number of axioms, as well as an appropriate induction principle, that constrain `Nat` to be freely generated by `zero` and `succ`. The axioms and the induction principle are automatically generated when the user defines the datatype.

The basic data values in Athena are terms and propositions. Terms are s-expressions built from declared function symbols such as `+` and `pi`, and from *variables*, written as `?`I for any identifier I. Thus `?x, (+ ?foo pi), (+ (+ ?x ?y) ?z)`, are all terms. The (most general) sort of a term is inferred automatically; the user does not have to annotate variables with their sorts. A proposition P is either a term of sort Boolean (say, `(< pi (+ ?x ?y))`); or an expression of the form `(not` P`)` or `(`$\odot$ P_1 P_2`)` for $\odot \in \{$`and`, `or`, `if`, `iff`$\}$; or `(`Q $x_1 \cdots x_n$ P`)` where $Q \in \{$`forall`, `exists`$\}$ and each x_i a variable. Athena also checks the sorts of propositions automatically using a Hindley-Milner-like type inference algorithm.

The user interacts with Athena via a read-eval-print loop. Athena displays a prompt `>`, the user enters some input (either a phrase to be evaluated or a top-level directive such as `define`, `assert`, `declare`, etc.), Athena processes the user's input, displays the result, and the loop starts anew.

The most fundamental concept in Athena is the *assumption base*—a finite set of propositions that are assumed to hold, representing our "axiom set" or "knowledge base". Athena starts out with the empty assumption base, which then gets incrementally augmented with the conclusions of the deductions that the user successfully evaluates at the top level of the read-eval-print loop. proposition can also be explicitly added into the global assumption base with the top-level directive `assert`.

An Athena deduction D is always evaluated in a given assumption base β—a finite set of propositions that are assumed to hold for the purposes of D. Evaluating D in β will either produce a proposition P (the "conclusion" of D in β), or else it will generate an error or will diverge. If D does produce a conclusion P, Athena's semantics guarantee $\beta \models P$, i.e., that P is a logical consequence of β. There are several syntactic forms that can be used for deductions.

The form `pick-any` introduces universal generalizations: `(pick-any` $I_1 \cdots I_n$ D`)` binds the names $I_1 \cdots I_n$ to fresh variables $v_1, \ldots, v_n$ and evaluates D. If D yields a conclusion P, the result returned by the entire `pick-any` is $(\forall\, v_1, \ldots, v_n)\, P$.

The form `assume` introduces conditionals: to evaluate `(assume` P D`)` in an assumption base β, we evaluate D in $\beta \cup \{P\}$. If that produces a conclusion Q, the conditional $P \Rightarrow Q$ is returned as the result of the entire `assume`. The form `(assume-let ((`I P`))` D`)` works like `assume`, but also lexically binds the name I to the hypothesis P within D.

The form `(dlet ((`I_1 D_1`)` $\cdots$ `(`I_n D_n`))` D`)` is used for sequencing and naming deductions. To evaluate such a deduction in β, we first evaluate D_1 in β to obtain a conclusion P_1. We then bind I_1 to P_1, insert P_1 into β, and continue with D_2. The conclusions P_i of the various D_i are thus incrementally added to the assumption base, becoming available as lemmas for subsequent use. The body D is then evaluated in $\beta \cup \{P_1, \ldots, P_n\}$, and its conclusion becomes the conclusion of the entire `dlet`.

Advanced Synthetic Characters, Evil, and E*

Selmer Bringsjord[1], Sangeet Khemlani[2], Konstantine Arkoudas[3], Chris McEvoy[4], Marc Destefano[5], Matthew Daigle[6]
Department of Cognitive Science[1−5]
Department of Computer Science[1,3,4]
Rensselaer AI & Reasoning Laboratory:[1−5]
http://www.cogsci.rpi.edu/research/rair/index.php
Rensselaer Polytechnic Institute (RPI)
Troy NY 12180 USA
{selmer,arkouk,mcevoc,khemls,destem}@rpi.edu
6: Dept. of Computer Science Vanderbilt University Nashville TN mdaigle@isis.vanderbilt.edu

Abstract

We describe our approach to building advanced synthetic characters, within the paradigm of logic-based AI. Such characters don't merely evoke *beliefs* that they have various mental properties; rather, they must actually *have* such properties. You might (e.g.) believe a standard synthetic character to be evil, but you would of course be wrong. An *advanced* synthetic character, however, can literally *be* evil, because it has the requisite desires, beliefs, and cognitive powers. Our approach is based on our RASCALS architecture, which uses simple logical systems (first-order ones) for low-level (perception & action) and mid-level cognition, and advanced logical systems (e.g., epistemic and deontic logics) for more abstract cognition. To focus our approach herein, we provide a glimpse of our attempt to bring to life one particular advanced synthetic character from the "dark side" — the evil character known simply as E. Building E entails that, among other things, we formulate an underlying logico-mathematical definition of evil, and that we manage to engineer both an appropriate presentation of E, and communication between E and humans. For presentation, which we only encapsulate here, we use several techniques, including muscle simulation in graphics hardware and approximation of subsurface scattering. For communication, we use our own new "proof-based" approach to Natural Language Generation (NLG). We provide an account of this approach.

The Dearth of AI in AI

There's an unkind joke — which made the rounds (e.g.) at the Fall 2004 AAAI Fall Symposium on Human-Level AI — about the need to create, within AI, a special interest group called 'AI'. This kind of cynicism springs from the not uncommon, and not totally inaccurate, perception that most of AI research is aimed at exceedingly narrow problems light years away from the cognitive capacities that distinguish human persons.[1]

Human-level AI is now so unusual that an entire upcoming issue of *AI Magazine* will be devoted to the subject — a bit odd, given that, at least when the field was young, AI's journal of record would have *routinely* carried papers on mechanizing aspects of human-level cognition. Seminal AI thinkers like Simon, Newell, Turing — these researchers didn't shy away from fighting to capture human-level intelligence in machine terms. But now their attitude seems moribund.

But gaming, simulation, and digital entertainment (and hereafter we refer simply to 'gaming' to cover this entire field/market), thankfully, are different: *ultimately* anyway, they call for at least the *appearance* of human-level AI (Bringsjord 2001). (On a case-by-case basis, as various games show (e.g., *The Sims* (Electronic Arts Inc. 2000)), a *non*-advanced character will of course do just fine.) Gaming doesn't strive just for a better SAT-based planner, or another tweak in a learning algorithm that doesn't relate in the least to human learning. A SAT planner doesn't constitute a virtual person. But that's precisely what we want in gaming, at least ultimately. And even in the short term we want characters that at least *seem* human. Methodologically speaking, gaming's best bet for characters that seem human is to bite the bullet and strive to engineer characters that have what it takes to *be* human. This, at least, is our strategy.

Gaming and Full-Blown Personhood

Now, there are various ways to get clearer about what gaming, at least in the long-term, needs when it comes to human-level intelligence. One way is to say simply that gaming needs artificial creatures which, behaviorally at any rate, satisfy one or more plausible proposed definitions of personhood in the literature. One such definition has been proposed by Bringsjord in (Bringsjord 1997). This definition essentially amounts to the view that x is a person if and only if x has the *capacity*

1. to "will," to make choices and decisions, set plans and projects — autonomously;
2. for consciousness, for experiencing pain and sorrow and happiness, and a thousand other emotions — love, passion, gratitude, and so on;
3. for *self*-consciousness, for being aware of his/her states of mind, inclinations, preferences, etc., and for grasping the concept of him/herself;

*The R&D described in this paper has been supported in part by much appreciated grants from AFRL-Rome and DARPA-IPTO.

[1] An endless source of confirming examples can be found in the pages of the *Machine Learning* journal. The dominant learning technique that you yourself employ in striving to learn is *reading*; witness what you're doing at the moment. Yet, a vanishingly small amount of R&D on learning is devoted to getting a computer program to learn by reading.

4. to communicate through a language;
5. to know things and believe things, and to believe things about what others believe, and to believe things about what others believe about one's beliefs (and so on);
6. to desire not only particular objects and events, but also changes in his or her character;
7. to reason (for example, in the fashion exhibited in the writing and reading of this very paper).

Unfortunately, this list is daunting, especially if, like us, you really and truly want to engineer a virtual person in the *short term*. A large part of the problem is consciousness, which we still don't know how to represent in third-person machine terms (Bringsjord 1998; Bringsjord 2001). But even if we leave aside consciousness, the rest of the attributes in the above list make for mighty tough challenges. In the section *"Making the Challenge of Personhood Tractable"* we shall retreat from this list to someting doable in the near term, guided by particular scenarios that make natural homes for E. But in the end, whatever appears on this list is an engineering target for us; in the long term we must confront each clause. Accordingly, in the section *"How Does E Talk?"* we explain how we are shooting for clause 4, communication. We have made progress on some of the other clauses, but there is insufficient space to present that progress herein. Clause 5 is one we believe we have pretty much satisfied, via the formalization and implementation given in (Arkoudas & Bringsjord 2005).[2]

Current State of the Art versus Computational Persons

Synthetic Characters in Gaming

What's being done now in gaming, relative to full-blown personhood, is clearly inadequate; this can be quickly seen by turning to some standard work: Figure 1 shows an array of synthetic characters from the gaming domain; these will be familiar to many readers.[3]

None of these creatures has anything close to the distinguishing features of personhood. Sustained treatments of synthetic characters and how to build them are similarly limited. For example, consider Figure 2, taken from (Champandard 2003).[4] As a mere FSA, there is no knowledge and belief, no reasoning, no declarative memories, and no linguistic capacity. In short, and this is perhaps a better way of putting the overall problem infecting todays's virtual characters, all of the cognitive capacities that distinguish human persons, according to the science of cognition (e.g., (Goldstein 2005)), are missing. Even the state of the art using cognitive architectures (e.g., SOAR) is primitive when stacked against full-blown personhood (Ritter *et al.* June 2002).

Figure 1: Sample Synthetic Characters

What About Synthetic Characters in Cutting Edge Research?

What about research-grade work on synthetic characters? Many researchers are working on synthetic characters, and have produced some truly impressive systems. However, all such systems, however much they *appear* to be human persons, aren't. We now consider three examples of such work, and show in each that the character architectures don't have the underlying cognitive content that is necessary for personhood.

REA

An agent developed by (Cassell *et al.* 1999) known as REA is an example of a successful, robust agent whose developers focused primarily on embodied conversation and the conversational interface. She is described as being an expert in the domain of real estate, and interactions with REA are both believable and informative.

REA, however, is representative of many of the industry's most successful agents in that she excels at content management, but fails to deliver rich emotive and cognitive functionality. REA, after all, cannot generate English from *arbitrary* underlying knowledge. Like many of her peers, REA's underlying cognitive capabilities are modeled in an ad-hoc fashion. Her personality is in no way defined; her interactions within a particular situation lack subtlety and depth. While she excels as a simulated character and a conversational agent, she is bereft of the rich cognitive content with which advanced synthetic characters must

[2] A preprint is available online at http://kryten.mm.rpi.edu/arkoudas.bringsjord.clima.crc.pdf.

[3] Worst to best, in our eyes: Top-left, The Legend of Zelda; SC spits text upon entering room. Top-right, Chrono Trigger; tree-branching conversations. Middle-left, Might & Magic VI (Shopkeepers). Middle-right, Superfly Johnson from Daikatana; behavior scripting, attempts to follow player and act as a sidekick (fails!). Bottom-left, Galatea – Interactive Fiction award winner for Best NPC of 2000 (text-based). Bottom-right, Sims 2. But even here, nothing like what our RASCALS architecture has is present.

[4] This is an excellent book, and it's used in our lab for building synthetic characters. But relative to the loftier goals of reaching *bona fide* personhood in artificial characters, there's clearly a lot of work to be done.

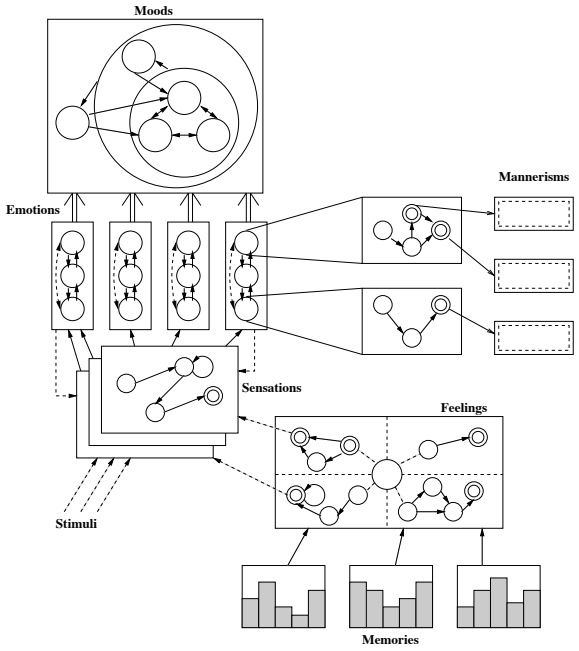

Figure 2: Impoverished Formalism for Synthetic Characters

behave.

The BEAT Architecture

In an engaging paper by (Gratch *et al.* 2002), an architecture is presented for developing rich synthetic characters. This architecture is known as the Behavior Expression Animation Toolkit Text-to-Nonverbal Behavior Module (BEAT). Under this architecture, emotion and cognitive content are produced systematically in a *simulation-based* approach.

Their simulation-based approach is built on top of *appraisal* theories of emotion, where emotions emerge from analysis of events and objects in a particular domain with respect to the agent's goals, standards, and attitudes. But as Gratch et al. themselves point out, appraisal theories "are rather vague about the assessment process...A promising line of research is integrating AI-based planning approaches, which might lead to a concretization of such theories." We will present the RASCALS paradigm as one that utilizes precisely the AI-based planning techniques Gratch et al. regard as promising.

Unfortunately, while Gratch et. al make wonderful advancements in the logistics of realizing agents, the issue of developing rich underlying cognitive content is eschewed. Even assuming that their simulation-based approach utilizes robust AI-based planning, the focus is not on developing true cognitive content but rather on its simulation and modeling.

Believable Interactive Embodied Agents

An approach more focused on building believable characters was proposed by (Pelachaud & Poggi 2002). They argue that research should include three distinct phases:

- *Phase 1: Empirical Research.* This phase involves research "aimed at finding out the regularities in the mind and behavior of Human Agents, and at constructing models of them."
- *Phase 2: Modeling Believable Interactive Embodied Agents.* Here, "rules are formalized, represented, and implemented in the construction of Agents."
- *Phase 3: Evaluation.* Finally, agents are tested on several levels, including "how well they fit the User's needs and how similar they look to a real Human Agent."

The "rule formalization" characterized in Phase 2 is, as Pelachaud and Poggi point out, indispensable when building believable characters. Since such rule formalizations are all expressible in first-order logic, their approach is actually a proper subset of the RASCALS approach. But formalizing and implementing rules is not enough to achieve true cognition; after all, cognition involves much more than simple rules/first-order logic. Iterated beliefs are beyond the reach of first-order logic. Finally, while Pelachaud and Poggi elaborate on linguistic rules and formalizations, they fail to mention anything about modeling cognition or interacting with a given knowledge base, and they make no remarks concerning the logistics behind rule formalization and implementation. The agents described therein all possess rudimentary cognitive content but come nowhere close to true cognitive or emotive capacity.

Making the Challenge of Personhood Tractable

How can we make the challenge of engineering a virtual person tractable in the very short term? Our lab has a two-part answer. First, assimilate everything out there regarding the *craft* of making viewers and users *believe* that the synthetic character they interact with is a genuine person. This is the same route that was followed by Bringsjord and Ferrucci in the design of the BRUTUS story generation system (Bringsjord & Ferrucci 2000). In a nutshell, B&F studied the literature on what responses are desired in readers by clever authors, and then reverse engineered back from these responses to a story generation system that triggers some of them. In connection with synthetic characters, this general strategy has impelled us to build up a large library on the design of synthetic charaters in stories and movies. In addition, we have built up a library of characters in film — specifically one that specializes in candidates for true evil. Within the space we have herein, however, this general strategy, and the results so far obtained, can't be presented. So we will settle here for a shortcut; it's the second part of our two-part answer. The shortcut is to work from concrete scenarios backwards by reverse engineering. We currently have two detailed scenarios under development. One is based on the evil people whose personalities are revealed in conversations in (Peck 1983); we leave this one aside for now. The second scenario, which is part of R&D undertaken in the area of wargaming, can be summarized as follows. (At the conference, we would provide a demo of conversation with E regarding both these scenarios, where that conversation

conforms to our account of evil; see *On our Formal Account of Evil*.)

E in Scenario 2, and Inference Therefrom

Let us imagine a man named simply E, a brutal warlord in a war-torn country. E is someone you're going to have to vanquish. He has moved up the ranks of the underworld in post-apocalyptic America after "success" in many, many murderous missions. E has taken a number of prisoners from an organization (let's call it simply O) he seeks to intimidate. O is chosen specifically because it is trying to rebuild the fractured US in the direction of a new federal governing[5]. Conforming to what has unfortunately become a gruesome pattern, E decides to film the beheading of one of these poor prisoners, and to release the video to O.

Given just this small amount of information, what can we infer about E's knowledge and reasoning? That it has at least the following six attributes:

1. *Mixed Representation*. E's knowledge is not simply linguistic or symbolic in nature. It includes visual or pictorial knowledge as well. For example, E clearly is thinking in terms of mental images, because he plans to gain leverage from the release of images and video. In addition, though it isn't pleasant to contemplate, E certainly has a "mental movie" that he knows he can turn into real life: he envisions how such executions work before performing them.
2. *Tapestried*. Presumably E's knowledge of his prisoners is relatively new. But this new knowledge is woven together with extensive prior knowledge and belief. For example, in E's case, he has extensive knowledge of O, and its principles regarding treatment of prisoners.
3. *Extreme Expressivity*. E's knowledge and reasoning requires highly expressive propositions. For example, he believes that O believes that it is universally forbidden to execute prisoners, and he believes that some of those aiding the United States' rebuilding effort will be struck with fear once the execution is complete and suitably publicized, and that that fear will affect their beliefs about what they should and shouldn't do.
4. *Mixed Inference Types*. E's reasoning is based not only on deductive inference, but also on educated guesses (abduction), and probabilistic inference (induction).
5. *Uses Natural Language*. E communicates in natural language, with his comrades, and with others as well.
6. *Multi-Agent Reasoning*. E is of course working in coordinated fashion with a number of accomplices, and to be effective, they must reason well as a group.

Working within the paradigm of logic-based AI (Bringsjord & Ferrucci 1998a; Bringsjord & Ferrucci 1998b; Nilsson 1991; Genesereth & Nilsson 1987), and using the MARMML knowledge representation and reasoning system, which is based on: the theory known as mental metalogic (Yang & Johnson-Laird 2000a; Yang & Johnson-Laird 2000b; Yang & Bringsjord 2005; Rinella, Bringsjord, & Yang 2001; Yang & Bringsjord 2001a; Yang & Bringsjord 2001b; Yang, Braine, & O'Brien 1998), the Denotational Proof Language known as Athena (Arkoudas 2000), Barwisean grids for diagrammatic knowledge and reasoning (see the mathematical section of (Barwise & Etchemendy 1995)), and RASCALS[6](see Figure 3), a revolutionary architecture for synthetic characters, we are building a virtual version of E that has the six attributes above.

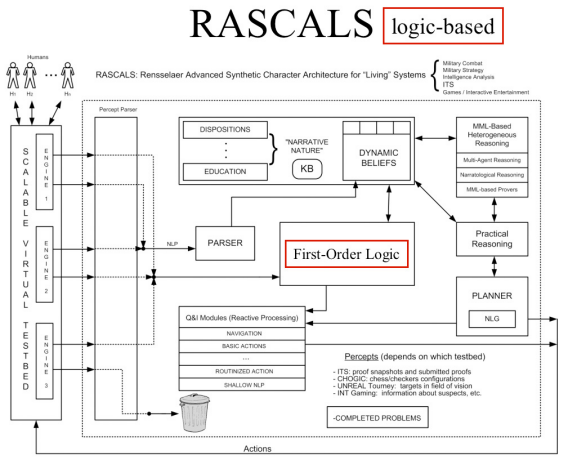

Figure 3: RASCALS: **R**ensselaer **A**dvanced **S**ynthetic **C**haracter **A**rchitecture for **L**ogical **S**ystems

Brief Remarks on the RASCALS Architecture

Let us say a few words about RASCALS, a brand new entry in the field of compuational cognitive modeling, which revolves around what are called *cognitive architectures* (e.g., SOAR (Rosenbloom, Laird, & Newell 1993); ACT-R (Anderson 1993; Anderson & Lebiere 1998; Anderson & Lebiere 2003); CLARION (Sun 2001); Polyscheme (Cassimatis 2002; Cassimatis *et al.* 2004)). What makes the RASCALS cognitive architecture distinctive? There is insufficient space here to convey any technical detail (for more details, see (Bringsjord forthcoming)); we make just three quick points about RASCALS, to wit:

- All other cognitive architectures we know of fall far short of the expressive power of RASCALS. For example, SOAR and ACT-R struggle to represent (let alone reason quickly over) textbook problems in logic (e.g., the Wise Man Problem = WMP) but in RASCALS such representations are effortless (see (Arkoudas & Bringsjord 2005) for the solution to WMP in Athena, included in RASCALS).
- The great challenge driving the field of computational cognitive modeling (CCM) is to unify *all* of human cognition; this challenge can be traced back to the birth of CCM in the work of Newell 1973. Such unification is achieved in one fell swoop by RASCALS, because *all* of cognition

[5]Coincidentally, we have recently learned that the game *Shattered World* for the X Box is related to our scenario.

[6]**R**ensselaer **A**dvanced **S**ynthetic **C**haracter **A**rchitecture for **L**ogical **S**ystems

can be formalized and mechanized in logic (though doing so requires some very complicated logics well beyond first-order logic, as in (Arkoudas & Bringsjord 2005)).

- While logic has been criticized as too slow for real-time perception-and-action-heavy computation, as you might see in first-person shooter (as opposed to a strategy game, which for obvious reasons fits nicely with the paradigm of logic-based AI), it has been shown that RASCALS is so fast that it can enable the real-time behavior of a mobile robot. We have shown this by having a logic-based mobile robot successfully navigate the wumpus world game, a staple in AI. (See Figures 4 and 5.)

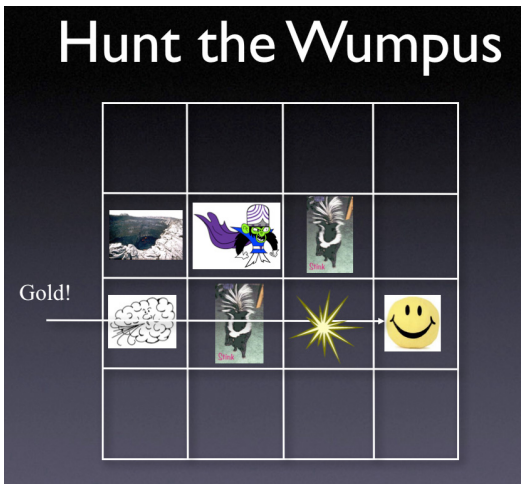

Figure 4: The Wumpus World Game

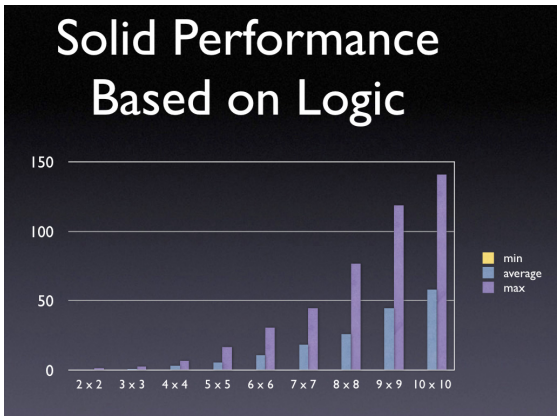

Figure 5: Performance of a RASCALS-Powered Robot in the Wumpus World

To show part of the underlying structure of E in connection with the attribute *Extreme Expressivity*, we now present an informal version of the formal account of evil that is implemented in our RASCALS architecture. This account specifically requires logics expressive enough to handle knowledge, belief, and ethical concepts. These logics go well beyond first-order logic; details and an implementation can be found in (Arkoudas & Bringsjord 2005). In the section *"E: The Presentation Level"* we explain the technology that allows E to speak naturally in English; that is, we show there part of the underlying structure of E associated with *Uses Natural Language*.

On our Formal Account of Evil

If we charitably push things in the direction of formally representing a definition of evil,[7] then we can understand Feinberg 2003 as advancing pretty much this definition:

Def 1 Person s is evil iff there exists some action a[8] such that

1. performing a is morally wrong;
2. s is morally blameworthy for performing a;
3. s's performing a causes considerable harm to others; and
4. the reasons or motives for s's performing a, along with "the elements that ground her moral blameworthiness," are unintelligible.

This is a decent starting place, but for us there are problems. For example, imagine that E invariably fails to cause *actual* harm. Surely he would still qualify as evil even if he were a bumbling villain. (If the knife slipped when he attempted decapitation, he would still be just as black-hearted.) This means that clause 3 should at least be replaced by

3′. s performs a in the hopes of causing considerable harm to others

But even this new definition, for reasons we don't have space to explain, is wholly inadequate. To give just a flavor for what E is currently based upon, we present simply our current best replacement for clause 4:

4″ were s a willing and open participant in the analysis of reasons and motives for s's seeking to perform a, it would be revealed that either

(i) these reasons and motives are unintelligible, or

(ii) s seeks to perform a in the service of goal g, and

(a) the anticipatable side-effects e of performing a are bad, but s cannot grasp this, or

(b) g itself is appraised as good by s when it is in fact bad.

Just this clause alone required much sustained analysis. (For a full chronicle of the evolution of a formally refined definition of betrayal from a rough starting one, see the chapter "Betrayal" in (Bringsjord & Ferrucci 2000).)

Keep in mind that this is still informal, kept that way in the interests of easing exposition. In the RASCALS-based implementation of E, evil must be expressed in purely formal form, which requires, again, that we use advanced logics of belief, knowledge, and obligation.[9]

[7] Feinberg's work is informal, and not suitable for direct use in AI and computer science.

[8] Or omission.

[9] For a look at the deontic logic (i.e., the logic of ethical concepts) we are relying upon, see (Horty 2001). Our mechanization

Keep in mind as well that we're not claiming that we have the perfect definition of evil. Some may object to our definition, and some of their objections may be trenchant. But the important point is to see how rich evil is — to see that it involves all kinds of highly cognitive powers and concepts that simply aren't found in today's synthetic characters. To be evil, one has to have beliefs, desires, and one has to have a lot of knowledge. The detailed configuration of these elements may not be exactly as we claim they ought to be, but no one can deny that the elements are needed. Without them, a synthetic character who is supposed to be evil is only a fake shell. And in the end, the shell will be revealed to be a shell: the illusion, at some point, will break down.

How Does E Talk?

As everyone knows, once the daunting challenge of rendering consciousness in computational terms is put aside, the greatest remaining challenge is that of giving an advanced synthetic character the power to communicate in a natural language (English, French, etc.) at the level of a human person. As you'll recall, communicative capacity is one of the clauses in the definition of personhood presented above. A plausible synthetic character must necessarily communicate in a fluid, robust manner. How, then, is such a rich form of communication implemented in E?

Reconciling Knowledge Representation and NLG

E speaks by parsing and processing formal knowledge; he develops an ontology based on internal and external queries, and then reasons over his knowledge to produce meaningful content. This content is then sent to his NLG module, translated into English, and finally presented to the user. Before we examine what goes on inside E's NLG module, let's take a moment to examine how E produces "meaningful content."

When we ask E a question, we are clearly interested in an answer that is both relevant and meaningful, an answer indistinguishable from those given by a real person. Assuming we have incomplete knowledge, suppose we ask of E, "Is John dangerous?" E approaches this question through formal logical analysis. The idea is to have E determine incontrovertibly whether John is dangerous or not. So, for instance, suppose E's knowledge base includes the following three facts:

1. DANGEROUS PEOPLE HAVE AUTOMATIC WEAPONS.
2. JOHN HAS A BERETTA AR-70 ASSAULT RIFLE.
3. THE BERETTA AR-70 ASSAULT RIFLE IS AN AUTOMATIC WEAPON.

None of the information above explicitly tells E whether John is dangerous or not, but clearly, when presented the above query, we want E to answer with an emphatic "Yes." Still, the answer itself is not enough. To ensure that E understands the nature of the question as well as the information he is dealing with, he must, upon request, provide *a justification for every answer*. The justification presented to the user is a formal proof, translated into English. Thus, E could answer as follows:

JOHN IS IN FACT DANGEROUS BECAUSE HE HAS A BERETTA AR-70 ASSAULT RIFLE. SINCE A BERETTA AR-70 ASSAULT RIFLE IS AN AUTOMATIC WEAPON, AND SINCE DANGEROUS PEOPLE HAVE AUTOMATIC WEAPONS, IT FOLLOWS THAT JOHN IS DANGEROUS.

Content is thus generated in the form of a formal proof. In general, the proofs generated will be more complex (they will use larger knowledge bases) and more sophisticated (they will use deontic and epistemic logic).

While the example is simple and rudimentary (that is, it makes use of only first-order logic and a small knowledge base), it demonstrates that E is taking heed of his knowledge to generate a meaningful reply. In the RASCALS architecture, answering "Yes" to the query above implies that E must in fact have the corresponding knowledge, an implication that does not hold for other architectures.

For a more formal method of analysis, we introduce the "Knowledge Code Test": If synthetic character C says something X or does something X designed to evoke in the mind of the human gamer/user the belief that C knows $P_1, P_2, \ldots$, then we should find a list of formulas, or the equivalent, corresponding to $P_1, P_2, \ldots$ in the code itself. The characters in Figure 1 would fail such a test, as would characters built on the basis of Champandard's specifications. An FSA, as a matter of mathematical fact, has no storage capability. A system with power that matches that of a full Turing machine is needed to pass the Knowledge Code Test (Lewis & Papadimitriou 1981).

But formal proofs are oftentimes too detailed to be of interest. Before we can even begin translating a proof into an English justification, we need verify that its level of abstraction is high enough that it is easy to read and understand. After all, formal natural deduction proofs are difficult and tedious to read. To represent proofs at a more wholistic, abstract level, we utilize the denotational proof language known as Athena (Arkoudas 2000). Athena is a programming language, development environment, and interactive proof system that evaluates and processes proofs as input. Its most prominent feature is its ability to present proofs in an abstract, top-level manner, isomorphic to that of a natural argument a human might use. By developing proofs in Athena at this level, a level high enough to be of interest to a human reader, we can be sure that the language generated from our NLG module is at precisely the level of abstraction we desire — neither too detailed nor too amorphous.

It's now time to look at precisely how English is generated from a formal proof.

Proof-based Natural Language Generation

Very few researchers are experimenting with the rigorous translation of formal proofs into natural language[10]. This is

of this this logic will be presented at the AAAI November 2005 Fall Symposium on Machine Ethics. The paper is available online at http://kryten.mm.rpi.edu/FS605ArkoudasAndBringsjord.pdf.

[10] An example of one such team is a research group at the University of Saarlande. The group had, until 1997, been developing

particularly odd when one considers the benefits of such a program. Natural deduction proofs, provided that they are developed in a sensible manner, are already poised for efficient translation. They require absolutely *no* further document structuring or content determination. That is, document planning, as defined by (Reiter & Dale 2000), is completely taken care of by using formal proofs in the first place.

Our NLG module receives as input a formal proof and returns as output English text. The English generated is an isomorph of the proof received. The structure of the justification, then, is precisely the same as the structure of the proof. If the justification uses *reductio ad absurdum* in the middle of the exposition, then you can be sure that there's a proof by contradiction in the middle of the formal proof.

Formal proofs are constructed from various different subproofs. A proof by contradiction is one such example of a type of subproof, but there are of course many others. Our system breaks a proof down to its constituent subproofs, translating each subproof from the top down. For example, assume the following:

1. CHICAGO IS A TARGET OR NEW YORK IS A TARGET
2. IF CHICAGO IS A TARGET, MILLIONS WILL DIE.
3. IF NEW YORK IS A TARGET, MILLIONS WILL DIE.

To deduce something meaningful from this information, we'll use a proof by cases. Our system translates this proof form as follows:

RECALL THAT CHICAGO OR NEW YORK IS A TARGET. EACH CASE PRODUCES THE SAME CONCLUSION; THAT IS, IF CHICAGO IS A TARGET THEN MILLIONS WILL DIE, AND IF NEW YORK IS A TARGET THEN MILLIONS WILL DIE. IT FOLLOWS THAT MILLIONS WILL DIE.

Predictably, documents produced in this manner, even when presented at a level abstract enough to make sense to a layperson, are rigid and, well, inhuman. They use the same phrases over and over again, they lack fluidity, and they are completely divorced of grace and wit. To boot, they disregard contextual information. Merely translating constituent subproofs to English will not produce natural English.

Nevertheless, this methodology provides a foundation for more sophisticated development. Once constituent subproofs are translated properly, they are sent to a microplanning system that maps particular subproofs to discourse relations (Hovy 1993). This mapping is known as a *message* and is not isomorphic. While the structure of the overall proof is preserved in the final document, individual subproofs are not treated with the same stringency. They can be molded and fitted to a number of different discourse relations for the sake of fluidity. Two more steps remain before natural language can be produced.

a system called PROVERB (Huang & Fiedler 1997). Their approach to proof-based translation was unique and extremely influential, though their project was largely unsuccessful.

Lexicalization is the process by which a lexicon of words is selected and mapped onto its symbolic counterparts. The content implicit in the proof, structured through subproof analysis and discourse relations, needs to be lexicalized before it can be presented as English text. That is, exact words and phrases must be chosen to represent relationships and predicates. For instance, TARGET(CHICAGO) must be translated to CHICAGO IS A TARGET and BERETTA(JOHN) must be translated to JOHN HAS A BERETTA before we can move on to gluing everything together. The only way this can happen is if a lexical database such as WordNet (Miller 1995) is augmented with domain-specific lexicalizations such as those specifying how to lexicalize "Beretta AR-70."

For even more fluidity, it's necessary to avoid referring to the same entities with the same phraseology. At the very least, pronouns should be substituted when referring to repeated concepts, persons, places, and objects. These substitutions are known as *referring expressions*, and need to be generated to truly produce fluid, humanlike English.

Fortunately, once the above issues are resolved, the information gathered therein can be plugged easily into a surface realizer such as KPML (Bateman 1997). In this fashion, proof-based NLG allows for the generation of both structured and expressive expositions.

E: The Presentation Level

To concretize our representation of evil (as in demos, e.g.; see the final section of the paper), we show E; a realistic real-time presentation of an evil talking head in the formal sense. In order to give E a realistic look, a range of facial expressions, and a flexible response to input, we simulate a subset of the muscles in the face. Each muscle in our model can contract, perturbing the underlying triangle mesh. Our simulation is based largely on that presented in (Waters 1987) and we have taken the approach of implementing the model almost entirely in a vertex shader. A parameterization for the tongue similiar to (King 2001) is used. A module for eye movements implements ideas presented in (Lee, Badler, & Badler 2002). Finally, we simulate subsurface scattering on the skin using the algorithm of (Sander, Gosselin, & Mitchell 2004). Our tool is shown in Figure 6.

Our Demos @ GameOn!

As mentioned above, at the conference we will allow attendees to discuss with E the two aforementioned scenarios, and this interaction will show our approach to the presentation level in action, and will manifest our formal account of evil in ordinary conversation that is based on our NLG technology.

References

[Anderson & Lebiere 1998] Anderson, J. R., and Lebiere, C. 1998. *The Atomic Components of Thought*. Mahwah, NJ: Lawrence Erlbaum.

[Anderson & Lebiere 2003] Anderson, J., and Lebiere, C. 2003. The newell test for a theory of cognition. *Behavioral and Brain Sciences* 26:587–640.

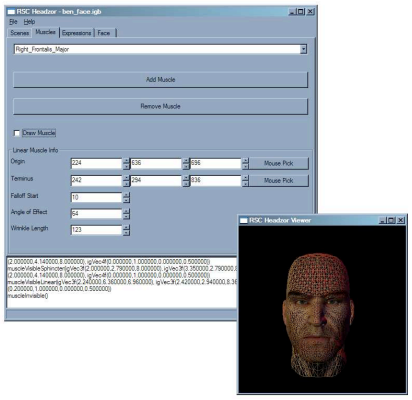

Figure 6: Tool for Manipulating Facial Muscles on E

[Anderson 1993] Anderson, J. R. 1993. *Rules of Mind*. Hillsdale, NJ: Lawrence Erlbaum.

[Arkoudas & Bringsjord 2005] Arkoudas, K., and Bringsjord, S. 2005. Metareasoning for multi-agent epistemic logics. In *Fifth International Conference on Computational Logic In Multi-Agent Systems (CLIMA 2004)*, volume 3487 of *Lecture Notes in Artificial Intelligence (LNAI)*. New York: Springer-Verlag. 111–125.

[Arkoudas 2000] Arkoudas, K. 2000. *Denotational Proof Languages*. Ph.D. Dissertation, MIT.

[Barwise & Etchemendy 1995] Barwise, J., and Etchemendy, J. 1995. Heterogeneous logic. In Glasgow, J.; Narayanan, N.; and Chandrasekaran, B., eds., *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. Cambridge, MA: MIT Press. 211–234.

[Bateman 1997] Bateman, J. A. 1997. Enabling technology for multilingual natural language generation: the kpml development environment. *Nat. Lang. Eng.* 3(1):15–55.

[Bringsjord & Ferrucci 1998a] Bringsjord, S., and Ferrucci, D. 1998a. Logic and artificial intelligence: Divorced, still married, separated...? *Minds and Machines* 8:273–308.

[Bringsjord & Ferrucci 1998b] Bringsjord, S., and Ferrucci, D. 1998b. Reply to Thayse and Glymour on logic and artificial intelligence. *Minds and Machines* 8:313–315.

[Bringsjord & Ferrucci 2000] Bringsjord, S., and Ferrucci, D. 2000. *Artificial Intelligence and Literary Creativity: Inside the Mind of Brutus, a Storytelling Machine*. Mahwah, NJ: Lawrence Erlbaum.

[Bringsjord 1997] Bringsjord, S. 1997. *Abortion: A Dialogue*. Indianapolis, IN: Hackett.

[Bringsjord 1998] Bringsjord, S. 1998. Chess is too easy. *Technology Review* 101(2):23–28.

[Bringsjord 2001] Bringsjord, S. 2001. Is it possible to build dramatically compelling interactive digital entertainment (in the form, e.g., of computer games)? *Game Studies* 1(1). This is the inaugural issue. Url: `http://www.gamestudies.org`.

[Bringsjord forthcoming] Bringsjord, S. forthcoming. The RASCALS cognitive architecture: Logic top to bottom. In Sun, R., ed., *The Handbook of Computational Cognitive Modeling*. Cambridge University Press.

[Cassell *et al.* 1999] Cassell, J.; Bickmore, T.; Billinghurst, M.; Campbell, L.; Chang, K.; Vilhjalmsson, H.; and Yan, H. 1999. Embodiment in conversational interfaces: Rea. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, 520–527. New York, NY, USA: ACM Press.

[Cassimatis *et al.* 2004] Cassimatis, N.; Trafton, J.; Schultz, A.; and Bugajska, M. 2004. Integrating cognition, perception and action through mental simulation in robots. In *Proceedings of the 2004 AAAI Spring Symposium on Knowledge Representation and Ontology for Autonomous Systems*.

[Cassimatis 2002] Cassimatis, N. 2002. *Polyscheme: A Cognitive Architecture for Integrating Multiple Representation and Inference Schemes*. Ph.D. Dissertation, Massachusetts Institute of Technology (MIT).

[Champandard 2003] Champandard, A. 2003. *AI Game Development*. Berkeley, CA: New Riders.

[Electronic Arts Inc. 2000] Electronic Arts Inc. 2000. *The Sims^{TM}: The People Simulator from the Creator of SimCity^{TM}*. Austin, TX: Aspyr Media.

[Feinberg 2003] Feinberg, J. 2003. *Problems at the Roots of Law*. New York, NY: Oxford University Press.

[Genesereth & Nilsson 1987] Genesereth, M., and Nilsson, N. 1987. *Logical Foundations of Artificial Intelligence*. Los Altos, CA: Morgan Kaufmann.

[Goldstein 2005] Goldstein, E. B. 2005. *Cognitive Psychology: Connecting Mind, Research, and Everyday Experience*. Belmont, CA: Wadsworth.

[Gratch *et al.* 2002] Gratch, J.; Rickel, J.; Andre, E.; Cassell, J.; Petajan, E.; and Badler, N. 2002. Creating interactive virtual humans: Some assembly required. *IEEE Intelligent Systems* 17(4):54–63.

[Horty 2001] Horty, J. 2001. *Agency and Deontic Logic*. New York, NY: Oxford University Press.

[Hovy 1993] Hovy, E. H. 1993. Automated discourse generation using discourse structure relations. *Artif. Intell.* 63(1-2):341–385.

[Huang & Fiedler 1997] Huang, X., and Fiedler, A. 1997. Proof verbalization as an application of NLG. In Pollack, M. E., ed., *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, 965–970. Nagoya, Japan: Morgan Kaufmann.

[King 2001] King, S. A. 2001. *A Facial Model and Animation Techniques for Animated Speech*. Ph.D. Dissertation, Ohio State University.

[Lee, Badler, & Badler 2002] Lee, S. P.; Badler, J. B.; and Badler, N. I. 2002. Eyes alive. *ACM Transactions on Graphics* 21(3):637–644.

[Lewis & Papadimitriou 1981] Lewis, H., and Papadimitriou, C. 1981. *Elements of the Theory of Computation*. Englewood Cliffs, NJ: Prentice Hall.

[Miller 1995] Miller, G. A. 1995. Wordnet: a lexical database for english. *Commun. ACM* 38(11):39–41.

[Newell 1973] Newell, A. 1973. You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In Chase, W., ed., *Visual Information Processing*. New York: Academic Press. 283–308.

[Nilsson 1991] Nilsson, N. 1991. Logic and Artificial Intelligence. *Artificial Intelligence* 47:31–56.

[Peck 1983] Peck, M. S. 1983. *People of the Lie*. New York, NY: Simon and Shuster.

[Pelachaud & Poggi 2002] Pelachaud, C., and Poggi, I. 2002. Multimodal embodied agents. *Knowl. Eng. Rev.* 17(2):181–196.

[Reiter & Dale 2000] Reiter, E., and Dale, R. 2000. *Building natural language generation systems*. New York, NY, USA: Cambridge University Press.

[Rinella, Bringsjord, & Yang 2001] Rinella, K.; Bringsjord, S.; and Yang, Y. 2001. Efficacious logic instruction: People are not irremediably poor deductive reasoners. In Moore, J. D., and Stenning, K., eds., *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*. Mahwah, NJ: Lawrence Erlbaum Associates. 851–856.

[Ritter *et al.* June 2002] Ritter, F.; Shadbolt, N.; Elliman, D.; Young, R.; Gobet, F.; and Baxter, G. June 2002. Techniques for modeling human performance in synthetic environments: A supplementary review. Technical report, Human Systems Information Analysis Center, Wright-Patterson Air Force Base, OH.

[Rosenbloom, Laird, & Newell 1993] Rosenbloom, P.; Laird, J.; and Newell, A., eds. 1993. *The Soar Papers: Research on Integrated Intelligence*. Cambridge, MA: MIT Press.

[Sander, Gosselin, & Mitchell 2004] Sander, P. V.; Gosselin, D.; and Mitchell, J. L. 2004. Real-time skin rendering on graphics hardware. In *Proceedings of ACM SIGGRAPH*.

[Sun 2001] Sun, R. 2001. *Duality of the Mind*. Mahwah, NJ: Lawrence Erlbaum Associates.

[Waters 1987] Waters, K. 1987. A muscle model for animating three-dimensional facial expression. In *Proceedings of ACM SIGGRAPH*, volume 21, 17–24.

[Yang & Bringsjord 2001a] Yang, Y., and Bringsjord, S. 2001a. Mental metalogic: A new paradigm for psychology of reasoning. In *Proceedings of the Third International Conference on Cognitive Science (ICCS 2001)*. Hefei, China: Press of the University of Science and Technology of China. 199–204.

[Yang & Bringsjord 2001b] Yang, Y., and Bringsjord, S. 2001b. The mental possible worlds mechanism: A new method for analyzing logical reasoning problems on the gre. In *Proceedings of the Third International Conference on Cognitive Science (ICCS 2001)*. Hefei, China: Press of the University of Science and Technology of China. 205–210.

[Yang & Bringsjord 2005] Yang, Y., and Bringsjord, S. 2005. *Mental Metalogic: A New, Unifying Theory of Human and Machine Reasoning*. Mahway, NJ: Erlbaum.

[Yang & Johnson-Laird 2000a] Yang, Y., and Johnson-Laird, P. N. 2000a. How to eliminate illusions in quantified reasoning. *Memory and Cognition* 28(6):1050–1059.

[Yang & Johnson-Laird 2000b] Yang, Y., and Johnson-Laird, P. N. 2000b. Illusory inferences with quantified assertions. *Memory and Cognition* 28(3):452–465.

[Yang, Braine, & O'Brien 1998] Yang, Y.; Braine, M.; and O'Brien, D. 1998. Some empirical justification of one predicate-logic model. In Braine, M., and O'Brien, D., eds., *Mental Logic*. Mahwah, NJ: Lawrence Erlbaum Associates. 333–365.

Toward Ethical Robots via Mechanized Deontic Logic*

Konstantine Arkoudas and **Selmer Bringsjord**
Rensselaer AI & Reasoning (RAIR) Lab
Department of Cognitive Science
Department of Computer Science
Rensselaer Polytechnic Institute (RPI)
Troy NY 12180 USA
{arkouk,selmer}@rpi.edu

Paul Bello
Air Force Research Laboratory
Information Directorate
525 Brooks Rd.
Rome NY 13441-4515
Paul.Bello@rl.af.mil

Abstract

We suggest that mechanized multi-agent deontic logics might be appropriate vehicles for engineering trustworthy robots. Mechanically checked proofs in such logics can serve to establish the permissibility (or obligatoriness) of agent actions, and such proofs, when translated into English, can also explain the rationale behind those actions. We use the logical framework Athena to encode a natural deduction system for a deontic logic recently proposed by Horty for reasoning about what agents ought to do. We present the syntax and semantics of the logic, discuss its encoding in Athena, and illustrate with an example of a mechanized proof.

Introduction

As machines assume an increasingly prominent role in our lives, there is little doubt that they will eventually be called upon to make important, ethically charged decisions. How can we trust that such decisions will be made on sound ethical principles? Some have claimed that such trust is impossible and that, inevitably, AI will produce robots that both have tremendous power and behave immorally (Joy 2000). These predictions certainly have some traction, particularly among a public that seems bent on paying good money to see films depicting such dark futures. But our outlook is a good deal more optimistic. We see no reason why the future, at least in principle, can't be engineered to preclude doomsday scenarios of malicious robots taking over the world.

One approach to the task of building well-behaved robots emphasizes careful ethical reasoning based on mechanized formal logics of action, obligation, and permissibility; that is the approach we explore in this paper. It is a line of research in the spirit of Leibniz's famous dream of a universal moral calculus (Leibniz 1984):

> When controversies arise, there will be no more need for a disputation between two philosophers than there would be between two accountants [computistas]. It would be enough for them to pick up their pens and sit at their abacuses, and say to each other (perhaps having summoned a mutual friend): 'Let us calculate.'

*We gratefully acknowledge that this research was in part supported by Air Force Research Labs (AFRL), Rome.

In the future we envisage, Leibniz's "calculation" would boil down to formal proof and/or model generation in rigorously defined, machine-implemented logics of action and obligation.

Such logics would allow for *proofs* establishing that:

1. Robots only take permissible actions; and
2. all actions that are obligatory for robots are actually performed by them (subject to ties and conflicts among available actions).

Moreover, such proofs would be highly reliable (i.e., have a very small "trusted base"), and explained in ordinary English.

Clearly, this remains largely a vision. There are many thorny issues, not least among which are criticisms regarding the practical relevance of such formal logics, efficiency issues in their mechanization, etc.; we will discuss some of these points shortly. Nevertheless, mechanized ethical reasoning remains an intriguing vision worth investigating.

Of course one could also object to the wisdom of logic-based AI in general. While other ways of pursuing AI may well be preferable in certain contexts, we believe that in this case a logic-based approach (Bringsjord & Ferrucci 1998a; 1998b; Genesereth & Nilsson 1987; Nilsson 1991; Bringsjord, Arkoudas, & Schimanski forthcoming) is promising because one of the central issues here is that of trust—and mechanized formal proofs are perhaps the single most effective tool at our disposal for establishing trust.

Deontic logic, agency, and action

In standard deontic logic (Chellas 1980; Hilpinen 2001; Aqvist 1984), or just SDL, the formula $\bigcirc P$ can be interpreted as saying that *it ought to be the case that P*, where P denotes some state of affairs or proposition. Notice that there is no agent in the picture, nor are there actions that an agent might perform. This is a direct consequence of the fact that SDL is derived directly from standard modal logic, which applies the possibility and necessity operators $\Diamond$ and $\Box$ to formulae standing for propositions or states of affairs. For example, the deontic logic D^* has one rule of inference, viz.,

$$\frac{P \rightarrow Q}{\bigcirc P \rightarrow \bigcirc Q}$$

and three axiom schemas:

- $(\bigcirc P \wedge \bigcirc Q) \rightarrow \bigcirc(P \wedge Q)$
- $\bigcirc \top$ ($\approx$ "That which must be is obligatory.")
- $\neg \bigcirc \bot$ ($\approx$ "Nothing impossible is obligatory.")

While D^* has some desirable properties, it and its relatives are plagued by various paradoxes (Hilpinen 2001), and, more importantly given present purposes, these logics aren't targeted at formalizing the concept of *actions* being obligatory (or permissible or forbidden) for an *agent*. Interestingly, deontic logics that have agents and their actions in mind do go back to the very dawn of this subfield of logic (von Wright 1951), but only recently has an "AI-friendly" semantics been proposed (Belnap, Perloff, & Xu 2001; Horty 2001) and corresponding axiomatizations been investigated (Murakami 2004).

We have used the Athena logical framework (briefly discussed in the next section) to encode a natural deduction calculus for a modern logic of agent action and obligation developed by Horty and axiomatized by Murakami in order to investigate mechanical deontic reasoning.

The ideal conditions for building "ethical robots" via a logic-based approach to AI would be as follows: we would have an expressive deontic logic $\mathcal{L}$ of high practical relevance, and an efficient algorithm for determining theoremhood in $\mathcal{L}$. That algorithm could then be built into the robot (perhaps implemented directly on its hardware), and the robot would only take an ethically charged action if it could formally prove that the action is permissible. Unfortunately, there is a legendarily strong tension between expressiveness and efficiency, and so it is certain that these ideal conditions will never obtain. For expressiveness, we will likely need highly hybrid modal and deontic logics that are at least first-order, which means that theoremhood in such logics will be undecidable. Even for decidable logics, such as the zero-order version of Horty's system [1] that we present in this paper, decision procedures are likely to be of inordinate computational complexity.

Therefore, we must reconcile ourselves to the possibility that a robot might not be able by itself to pass judgment on certain actions that it is contemplating; and that instead of a single monolithic decision procedure for deontic theoremhood (or validity, assuming the logic is complete), the robot might instead need to be armed with a knowledge base of lemmas and a panoply of *tactics*, each capable of settling certain restricted subclasses of deontic questions. This might well turn out to be sufficient in practice. If and when a complicated proposition arises that ethically paralyzes the robot, humans could intervene to settle the situation as they see fit. The logical framework that we have used to mechanize Horty's logic, Athena (Arkoudas 2003), facilitates the formulation of lemmas and of highly reliable tactics.

Athena

Athena (Arkoudas 2003) is a new interactive theorem proving system for polymorphic multi-sorted first-order logic that incorporates facilities for model generation, automated theorem proving, and structured proof representation and checking. It also provides a higher-order functional programming language, and a proof abstraction mechanism for expressing arbitrarily complicated inference *methods* in a way that guarantees soundness, akin to the tactics and tacticals of LCF-style systems such as HOL (Gordon & Melham 1993) and Isabelle (Paulson 1994). Proof automation is achieved in two ways: first, through user-formulated proof methods; and second, through the seamless integration of state-of-the-art ATPs such as Vampire (Voronkov 1995) and Spass (Weidenbach 2001) as primitive black boxes for general reasoning. For model generation, Athena integrates Paradox (Claessen & Sorensson 2003), a new highly efficient model finder. For proof representation and checking, Athena uses a block-structured Fitch-style natural deduction calculus (Pelletier 1999) with novel syntactic constructs and a formal semantics based on the abstraction of *assumption bases* (Arkoudas 2000). Most interestingly, a block-structured natural deduction format is used not only for writing proofs, but also for writing tactics (methods). This is a novel feature of Athena. Tactics in this style are considerably easier to write and remarkably useful in making proofs more modular and abstract.

Athena has been used to implement a proof-emitting optimizing compiler (Rinard & Marinov 1999); to integrate model checking and theorem proving for relational reasoning (Arkoudas *et al.* 2003); to implement various "certifying" algorithms (Arkoudas & Rinard 2004); to verify the core operations of a Unix-like file system (Arkoudas *et al.* 2004); to prove the correctness of dataflow analyses (Hao 2002); and to reason about generic software (Musser 2004). A concise presentation of Athena's syntax and semantics can be found elsewhere (Arvizo 2002).

Horty's logic and its Athena encoding

Murakami (Murakami 2004) presents an axiomatization of Horty's utilitarian formulation of multi-agent deontic logic (Horty 2001), and shows it decidable by proving that it has the finite model property. In this section we develop an alternative, sequent-based natural-deduction formulation of Murakami's system. The logic is encoded in Athena, which is then used as a metalanguage in order to reason about the encoded object language; we have used this methodology successfully with other intensional logics (Arkoudas & Bringsjord 2005). In what follows we briefly review the abstract syntax and semantics of the logic, and then present our formulation of a natural deduction system for it.

We use the letters P, Q, $R, \ldots$, to designate arbitrary *propositions*, built according to the following abstract grammar:

$$\begin{array}{rcl} P & ::= & A \mid \top \mid \bot \mid \neg P \mid P \wedge Q \mid P \vee Q \mid P \Rightarrow Q \\ & \mid & \Box P \mid \Diamond P \mid [\alpha \textit{ cstit: } P] \mid \odot [\alpha \textit{ cstit: } P] \end{array}$$

where A and α range over a countable set of atomic propositions ("atoms") and a primitive domain of *agents*, respectively. Propositions of the form $[\alpha \textit{ cstit: } P]$ and $\odot [\alpha \textit{ cstit: } P]$ are read as "α sees to it that P" and "α

[1] Proved complete and decidable by Murakami (Murakami 2004).

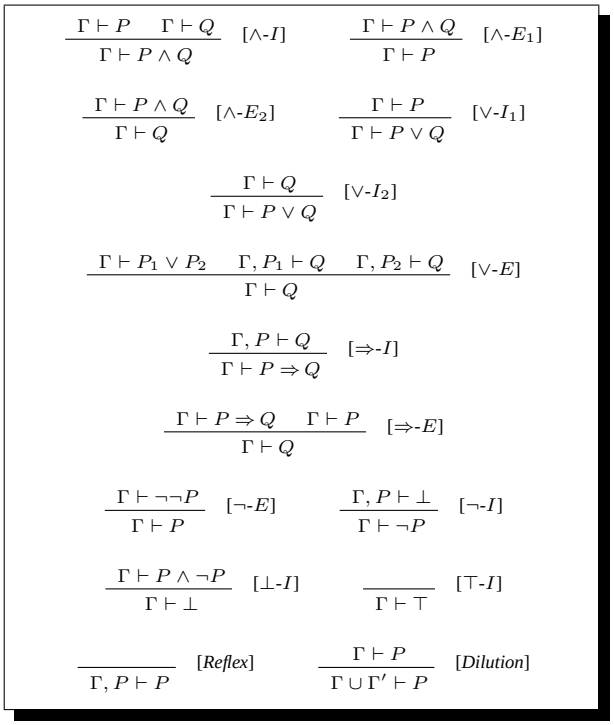

Figure 1: Inference rules for the propositional connectives.

ought to see to it that P," respectively. [2] We stress that $\odot\,[\alpha$ *cstit:* $P]$ is *not* read as "It ought to be the case that α sees to it that P." That is the classic Meinong-Chisholm "ought-to-be" analysis of agency, captured by another formula altogether, $\bigcirc\,[\alpha$ *cstit:* $P]$, where $\bigcirc$ is the non-agent-oriented "ought" operator similar to what is found in SDL. In Horty's semantics, $\bigcirc\,[\alpha$ *cstit:* $P]$ and $\odot\,[\alpha$ *cstit:* $P]$ are not equivalent statements; neither implies the other (Horty 2001). In general, the operator $\bigcirc$, taken over from SDL, applies to P just in case P holds in each of the best worlds. As Horty explains, an analogue to this basic idea is expressed by $\odot\,[\alpha$ *cstit:* $P]$, because this locution holds whenever P is ensured by each of the agent's best actions. (We have little use for the standard obligation operator $\bigcirc$ and hence we omit it from our formulation, although it could be easily included.)

The formal semantics are given on the basis of the theory of indeterminate branching time (Prior 1967; Thomason 1984), augmented with constructs for dealing with agent actions. The usual Kripke frames of modal logic are replaced by *deontic stit frames*. A deontic stit frame has the following components:

- A set of *moments* M, along with a strict partial order $<$ on M (i.e., $<$ is irreflexive and transitive, and hence asymmetric as well). A maximal linearly ordered subset of M is called a *history*. The set of all histories containing a moment $m \in M$ is written as H_m.
- A set A of *agents*.
- A binary function *Choice* that maps any given agent α and moment m into a partition *Choice*(α, m) of H_m. This function must satisfy two constraints: *independence of agents*, and *no choice between undivided histories*; see (Horty 2001) for details.
- For each $m \in M$, a utility function V_m from H_m into some partially ordered set of values (typically the real numbers).

The semantics are given with respect to moment/history pairs. Specifically, a *deontic stit model* is a deontic stit frame along with a truth valuation that maps each pair $\langle m, h\rangle$ with $m \in M, h \in H_m$ into a subset of atomic propositions (intuitively, these are the atoms that are true at the index $\langle m, h\rangle$). Given a deontic stit model $\mathcal{M}$ and a moment/history pair $\langle m, h\rangle$ (with $h \in H_m$), we write $\mathcal{M} \models_{\langle m, h\rangle} P$ to mean that $\mathcal{M}$ satisfies proposition P at index $\langle m, h\rangle$. The definition of $\mathcal{M} \models_{\langle m, h\rangle} P$ is given by induction on the structure of P. The cases of atoms and propositional combinations are standard. Cstit propositions are handled as follows: $\mathcal{M} \models_{\langle m, h\rangle} [\alpha$ *cstit:* $P]$ iff

$$\mathcal{M} \models_{\langle m, h'\rangle} P$$

for every $h' \in$ *block*(m, α, h), where *block*(m, α, h) is the unique block (equivalence class) containing h in the partition *Choice*(α, m) of H_m. We refer the reader to (Horty 2001) for the semantics of $\odot\,[\alpha$ *cstit:* $P]$.

A *sequent* $\Gamma \vdash P$ consists of a context Γ (a finite set of propositions) and a proposition P. Intuitively, this states that P follows from Γ. We write Γ, P (or P, Γ) as an abbreviation for $\Gamma \cup \{P\}$. The sequent calculus that we use consists of a collection of inference rules for deriving judgments of the form $\Gamma \vdash P$. Figure 1 shows the inference rules that deal with the standard propositional connectives. These are the usual introduction and elimination rules for each connective, in addition to reflexivity and dilution (weakening). Further, we have thirteen rules pertaining to the modal and deontic operators, shown in Figure 2. $[R_1]$, $[R_4]$ and $[R_6]$ are sequent formulations of Kripke's "K" axiom for the operators $\Box$, *cstit*, and $\odot$, respectively. $[R_2]$ and $[R_7]$ are the usual "T" axioms of modal logic for $\Box$ and *cstit*. $[R_3]$ is the "axiom 5" for $\Box$. $[R_8]$ and $[R_9]$ express that necessary truths are ensured (if P is necessary then every agent sees to it) and obligatory. $[R_{10}]$ asserts that obligations are possible. $[R_{12}]$ is a necessitation rule ensuring that all tautologies (propositions derivable from the empty context) are necessary. (A similar necessitation rule for *cstit* can be derived from $[R_8]$ in tandem with $[R_{12}]$, so we do not need to take it as primitive.) $[R_{13}]$ says that if α seeing to P strictly implies α seeing to Q, then if α ought to stit P then α also ought to stit Q. Finally, $[R_5]$ is a slightly disguised formulation of the standard "axiom 5" for *cstit*. It is provably equivalent to

$$\neg[\alpha \textit{ cstit: } \neg P] \Rightarrow [\alpha \textit{ cstit: } \neg[\alpha \textit{ cstit: } \neg P]]$$

[2] The 'c' in *cstit* stands for "Chellas." Horty (Horty 2001) attributes the naming to the fact that *cstit* is analogous—though not identical—to an operator introduced by Brian Chellas in his 1969 doctoral dissertation (Chellas 1969). There are other stit operators in the literature, e.g., the achievement stit ("astit"), the deliberative stit ("dstit"), etc.

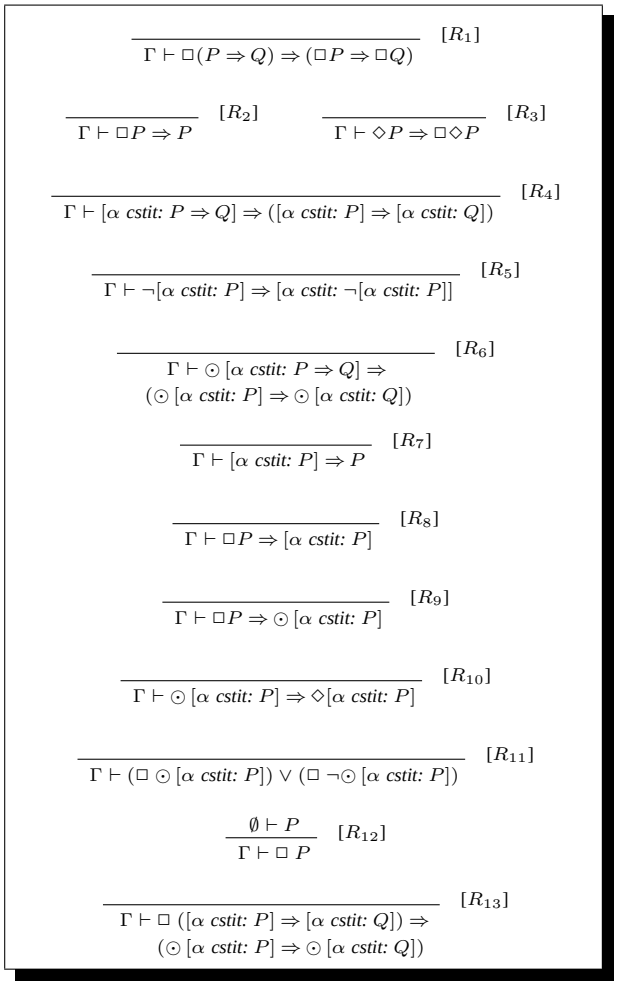

Figure 2: Inference rules for the deontic operators.

which is of the exact same form as the "axiom 5":

$$\Diamond P \Rightarrow \Box \Diamond P$$

once we realize that $\Diamond P$ stands for $\neg\Box \neg P$.

Our Athena formalization introduces a domain of agents and a datatype that captures the abstract syntax of the propositions of the logic:

```
(datatype Prop
   False
   True
  (Atom Boolean)
  (If Prop Prop)
  (Not Prop)
  (And Prop Prop)
  (Or Prop Prop)
  (Stit Agent Prop)
  (OughtToStit Agent Prop)
  (Nec Prop)
  (Pos Prop))
```

We proceed to introduce a binary relation `sequent` that may obtain between a finite set of propositions and a single proposition:

```
(declare sequent (-> ((FSet-Of Prop) Prop)
                     Boolean))
```

Here `FSet-Of` is a unary sort constructor: for any sort `T`, `(FSet-Of T)` is a new sort representing the set of all finite sets of elements of `T`. Finite sets are built with two polymorphic constructors: the constant `null`, representing the empty set; and the binary constructor `insert`, which takes an element x of sort `T` and a finite set S (of sort `(FSet-Of T)`) and returns the set $\{x\} \cup S$. We also have all the usual set-theoretic operations available (`union`, `intersection`, etc.).

The intended interpretation is that if `(sequent` S P`)` holds for a set of propositions S and a proposition P, then the sequent $S \vdash P$ is derivable in the logic via the above rules. Accordingly, we introduce axioms capturing those rules. For instance, the conjunction introduction rule and rule $[R_{10}]$ are represented, respectively, by the following two axioms:

```
(define And-I
  (forall ?Gamma ?P ?Q
    (if (and (sequent ?Gamma ?P)
             (sequent ?Gamma ?Q))
        (sequent ?Gamma (And ?P ?Q)))))

(define R10
  (forall ?Gamma ?a ?P
    (sequent ?Gamma
       (If (OughtToStit ?a ?P)
           (Pos (Stit ?a ?P))))))
```

Note the object/meta-level distinction between, e.g., `and` and `And`. The former is a native Athena propositional constructor, i.e., part of the metalogic, whereas the latter is a propositional constructor of the encoded object logic.

As we have argued elsewhere (Arkoudas & Bringsjord 2005), such a direct proof-theoretic encoding of a modal logic in a first-order logical framework such as Athena carries several advantages:

- The proofs are in natural deduction format and hence easier to read, write, and translate into English.
- Theorem proving is facilitated because we are able to leverage state-of-the-art automated theorem provers (ATPs) such as Vampire (Voronkov 1995) and Spass (Weidenbach 2001) that are integrated with Athena. Tactics can be programmed at a fairly high level of abstraction, with tedious details outsourced to the ATPs.
- Because we have explicitly encoded the abstract syntax of the logic, we are able to quantify over agents, propositions, and sequents. This provides us with the generalization benefits of higher-order logic, even though we are working in a first-order system.

Example

As a simple but non-trivial example, we present the Athena proof of the following "iterated *cstit*" result:

$$[\alpha \textit{ cstit: } P] \Rightarrow [\alpha \textit{ cstit: } [\alpha \textit{ cstit: } P]] \qquad (1)$$

```
pick-any P a
 begin
   S1 := (sequent
           null
           (If (Not (Stit a (Not (Not P))))
               (Stit
                 a
                 (Not (Stit
                        a (Not (Not P)))))))
         from R5;
    S2 := (sequent
            null
            (If (Not (Stit
                       a
                       (Not (Stit
                              a
                              (Not (Not P))))))
                (Not (Not (Stit
                            a
                            (Not (Not P)))))))
          from S1, contrapositive;
    S3 := prove (sequent
                  null
                  (If (Not
                        (Not (Stit
                               a
                               (Not (Not P)))))
                      (Stit a (Not (Not P))))));
    S4 := (sequent
            null
            (If (Not (Stit
                       a
                       (Not (Stit
                              a
                              (Not (Not P))))))
                (Stit a (Not (Not P)))))
          from S2, S3, transitivity;
    S5 := prove (sequent
                  null (Iff P (Not (Not P)))));
    S6 := (sequent null
            (Iff
              (Not (Stit a (Not (Stit a P))))
              (Not (Stit
                     a
                     (Not (Stit
                            a
                            (Not (Not P))))))))
          from S5, lemma-1.7;
```

Figure 3: Athena proof of Lemma 1.8, part 1.

The proof is easily turned into a tactic that can be applied to any given agent and proposition.

A number of lemmas are used in the proof. Most of them express straightforward propositional logic tautologies and are proved automatically by outsourcing them to the ATPs that are integrated with Athena. For instance, the first four lemmas below respectively express the transitivity of logical implication, the contrapositive law, the cut, and disjunctive syllogism.

Lemma 1.1 *If* $\Gamma \vdash P \Rightarrow Q$ *and* $\Gamma \vdash Q \Rightarrow R$ *then* $\Gamma \vdash P \Rightarrow R$.

```
   S7 := (sequent
           null
           (If (Not (Stit a (Not (Stit a P))))
               (Not (Stit
                      a
                      (Not (Stit
                             a
                             (Not (Not P))))))))
         from S6, Iff-Elim-1;
   S8 := prove
           (sequent null (If (Not (Not P)) P));
   S9 := (sequent
           null
           (If (Stit a (Not (Not P)))
               (Stit a P)))
         from S8, lemma-1.6;
 (sequent null
           (If (Not (Stit a (Not (Stit a P))))
               (Stit a P)))
 from S4, S7, S9, lemma-1.5
end
```

Figure 4: Athena proof of Lemma 1.8, part 2.

Lemma 1.2 *If* $\Gamma \vdash P \Rightarrow Q$ *then* $\Gamma \vdash \neg Q \Rightarrow \neg P$.

Lemma 1.3 *If* $\Gamma_1 \vdash P$ *and* $\Gamma_2, P \vdash Q$ *then* $\Gamma_1 \cup \Gamma_2 \vdash Q$.

Lemma 1.4 $\Gamma \vdash (P_1 \vee P_2) \Rightarrow (\neg P_2 \Rightarrow P1)$.

Lemma 1.5 *If* $\Gamma \vdash P' \Rightarrow Q'$, $\Gamma \vdash P \Rightarrow P'$, *and* $\Gamma \vdash Q' \Rightarrow Q$ *then* $\Gamma \vdash P \Rightarrow Q$.

A few properly deontic lemmas are also necessary:

Lemma 1.6 *For all agents* α *and propositions* P *and* Q, *if* $\emptyset \vdash P \Rightarrow Q$ *then* $\emptyset \vdash [\alpha \textit{ cstit: } P] \Rightarrow [\alpha \textit{ cstit: } Q]$.

Lemma 1.7 *For all agents* α *and propositions* P *and* Q, *if* $\emptyset \vdash P \Leftrightarrow Q$ *then* $\emptyset \vdash \neg[\alpha \textit{ cstit: } P] \Leftrightarrow \neg[\alpha \textit{ cstit: } Q]$.

Lemma 1.8 $\emptyset \vdash \neg[\alpha \textit{ cstit: } \neg[\alpha \textit{ cstit: } P]] \Rightarrow [\alpha \textit{ cstit: } P]$ *for all* α *and* P.

Lemma 1.9 $\emptyset \vdash P \Rightarrow \neg[\alpha \textit{ cstit: } \neg P]$.

Lemma 1.6, Lemma 1.7, and Lemma 1.9 are proved automatically. Lemma 1.8 is more challenging and requires user guidance. Its proof, in Athena's natural deduction system, is shown in two parts in Figure 3 and in Figure 4. Very brief explanations of the pertinent Athena constructs are given below to help the reader follow the code. For a more thorough treatment we refer the reader to the Athena Web site.

An Athena deduction D is always evaluated in a given *assumption base*—a finite set of propositions that are assumed to hold for the purposes of D. An assumption base thus represents our "axiom set" or "knowledge base." Athena starts out with the empty assumption base, which then gets incrementally augmented with the conclusions of the deductions that the user successfully evaluates. Propositions can also be explicitly added into the global assumption base with the top-level directive `assert`.

Evaluating a deduction D in an assumption base β will either produce a proposition F (the "conclusion" of D in β), or else it will generate an error or will diverge. If D does produce a conclusion F, Athena's semantics guarantee $\beta \models F$, i.e., that F is a logical consequence of β. There are several syntactic forms that can be used for deductions; they form a complete proof system for polymorphic multi-sorted first-order logic.

The form `pick-any` introduces universal generalizations: `pick-any` $I_1 \cdots I_n$ `begin` D `end` (for arbitrary subdeduction D) binds the names $I_1 \cdots I_n$ to fresh variables $v_1, \ldots, v_n$ and evaluates D. If and when D yields a conclusion F, the result returned by the entire `pick-any` is $\forall\, v_1, \ldots, v_n\, .\, F$.

The body of the proof is a semicolon-separated sequence of steps of the form

$$I_1 \texttt{ := } D_1 \texttt{;} \cdots \texttt{;} I_n \texttt{ := } D_n$$

where I_j is a name (identifier) and D_j an arbitrary subproof. The sequence is evaluated by recursively evaluating each D_j in turn, $j = 1, 2, \ldots$, obtaining a conclusion F_j, binding the name I_j to F_j, inserting F_j in the assumption base, and then proceeding with the next step, I_{j+1} `:=` D_{j+1}. The conclusion of the entire sequence is the conclusion of the last step, D_n. Note that the last step is not named.

A common proof step is of the form F `from` $F_1, \ldots, F_k$. This instructs Athena to try to automatically derive the conclusion F from the given premises $F_1, \ldots, F_k$ (all k of which must be in the assumption base). After performing some internal translations, Athena outsources this step to an ATP. If the ATP manages to solve the problem within a certain time limit (currently preset to a maximum of 60 seconds), then F is returned as the result of the step; otherwise an error message appears.

A similar step is of the form `prove` F. This attempts to automatically derive F from all the elements of the current assumption base. This is therefore equivalent to F `from` $F_1, \ldots, F_k$, where $F_1, \ldots, F_k$ are all and only the members of the current assumption base.

With the above lemmas at hand, the original goal can be proved as shown in Figure 5.

```
pick-any P a
 begin
  S1 := (sequent null
                 (If (Stit a P)
                     (Not (Stit a (Not (Stit a P))))))
        from lemma-1.9;
  S2 := (sequent null
                 (If (Not (Stit a (Not (Stit a P))))
                     (Stit a (Not
                               (Stit a (Not
                                        (Stit a P))))))
        from R5;
  S3 := (sequent
          null
          (If (Stit a P)
              (Stit a (Not
                        (Stit a (Not (Stit a P)))))))
          from S1, S2, transitivity;
  S4 :=  (sequent null
                  (If (Not (Stit a (Not (Stit a P))))
                      (Stit a P)))
         from lemma-1.8;
  S5 :=  (sequent
           null
           (If (Stit a (Not (Stit a (Not (Stit a P)))))
               (Stit a (Stit a P))))
         from S4, lemma-1.6;
 (sequent null (If (Stit a P)
                   (Stit a (Stit a P))))
 from S3, S5, transitivity
end
```

Figure 5: Athena proof of (1).

Conclusions

We have reported ongoing work on the mechanization of multi-agent logics of action and obligation. It is reasonable to believe that such logics might prove useful in engineering machines that can reason about what they ought to do. We presented an Athena implementation of a natural deduction calculus for a recently developed deontic logic of agency based on indeterminate branching-time semantics augmented with dominance utilitarianism, and presented an example of a mechanized proof in that system. We are currently using mechanized deontic logics to represent wargaming scenarios and to implement wargame agents capable of reasoning about their own ethical codes as well as those of their adversaries. In that direction, we plan to investigate the mechanization of defeasible deontic logics that allow for explicit modeling of contrary-to-duty obligations and violations (Van Der Torre 1997).

References

Aqvist, E. 1984. Deontic logic. In Gabbay, D., and Guenthner, F., eds., *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*. Dordrecht, The Netherlands: D. Reidel. 605–714.

Arkoudas, K., and Bringsjord, S. 2005. Metareasoning for multi-agent epistemic logics. In *Fifth International Conference on Computational Logic In Multi-Agent Systems (CLIMA 2004)*, volume 3487 of *Lecture Notes in Artificial Intelligence (LNAI)*. New York: Springer-Verlag. 111–125.

Arkoudas, K., and Rinard, M. 2004. Deductive runtime certification. In *Proceedings of the 2004 Workshop on Runtime Verification*, 39–56.

Arkoudas, K.; Khurshid, S.; Marinov, D.; and Rinard, M. 2003. Integrating model checking and theorem proving for relational reasoning. In *Seventh International Seminar on Relational Methods in Computer Science (RelMiCS 2003)*, volume 3015 of *Lecture Notes in Computer Science (LNCS)*, 21–33.

Arkoudas, K.; Zee, K.; Kuncak, V.; and Rinard, M. 2004. Verifying a file system implementation. In *Sixth International Conference on Formal Engineering Methods (ICFEM'04)*, volume 3308 of *Lecture Notes in Computer Science (LNCS)*, 373–390.

Arkoudas, K. 2000. *Denotational Proof Languages*. Ph.D. Dissertation, MIT, Department of Computer Science, Cambridge, USA.

Arkoudas, K. 2003. Athena. `http://www.pac.csail.mit.edu/athena`.

Arvizo, T. 2002. A virtual machine for a type-ω denotational proof language. Masters thesis, MIT, June 2002.

Belnap, N.; Perloff, M.; and Xu, M. 2001. *Facing the future*. Oxford University Press.

Bringsjord, S., and Ferrucci, D. 1998a. Logic and artificial intelligence: Divorced, still married, separated...? *Minds and Machines* 8:273–308.

Bringsjord, S., and Ferrucci, D. 1998b. Reply to Thayse and Glymour on logic and artificial intelligence. *Minds and Machines* 8:313–315.

Bringsjord, S.; Arkoudas, K.; and Schimanski, B. forthcoming. Logic-based AI for the new millennium. *AI Magazine*.

Chellas, B. 1969. The logical form of imperatives. PhD dissertation, Stanford Philosophy Department.

Chellas, B. F. 1980. *Modal Logic: an Introduction*. Cambridge University Press.

Claessen, K., and Sorensson, N. 2003. New techniques that improve Mace-style finite model building. In *Model Computation—principles, algorithms, applications*.

Genesereth, M., and Nilsson, N. 1987. *Logical Foundations of Artificial Intelligence*. Los Altos, CA: Morgan Kaufmann.

Gordon, M. J. C., and Melham, T. F. 1993. *Introduction to HOL, a theorem proving environment for higher-order logic*. Cambridge, England: Cambridge University Press.

Hao, M. 2002. Using a denotational proof language to verify dataflow analyses. Masters thesis, MIT, September 2002.

Hilpinen, R. 2001. Deontic Logic. In Goble, L., ed., *Philosophical Logic*. Oxford, UK: Blackwell. 159–182.

Horty, J. 2001. *Agency and Deontic Logic*. Oxford University Press.

Joy, W. 2000. Why the Future Doesn't Need Us. *Wired* 8(4).

Leibniz. 1984. *Notes on analysis*. Past Masters: Leibniz. Oxford University Press. Translated by George MacDonald Ross.

Murakami, Y. 2004. Utilitarian deontic logic. In *Proceedings of the Fifth International Conference on Advances in Modal Logic (AiML 2004)*, 288–302.

Musser, D. 2004. Generic Software Design. `www.cs.rpi.edu/~musser/gsd`.

Nilsson, N. 1991. Logic and Artificial Intelligence. *Artificial Intelligence* 47:31–56.

Paulson, L. 1994. *Isabelle, A Generic Theorem Prover*. Lecture Notes in Computer Science. Springer-Verlag.

Pelletier, F. J. 1999. A Brief History of Natural Deduction. *History and Philosophy of Logic* 20:1–31.

Prior, A. 1967. *Past, present, and future*. Oxford University Press.

Rinard, M., and Marinov, D. 1999. Credible compilation with pointers. In *Proceedings of the 1999 Workshop on Run-Time Result Verification*.

Thomason, R. 1984. Combinations of tense and modality. In Gabbay, D., and Guenthner, F., eds., *Extensions of classical logic*, volume 2 of *Handbook of Philosophical Logic*. Dordrecht, The Netherlands: D. Reidel.

Van Der Torre, L. 1997. *Reasoning about obligations*. Ph.D. Dissertation, Erasmus University, Rotterdam, Netherlands.

von Wright, G. 1951. Deontic logic. *Mind* 60:1–15.

Voronkov, A. 1995. The anatomy of Vampire: implementing bottom-up procedures with code trees. *Journal of Automated Reasoning* 15(2).

Weidenbach, C. 2001. Combining superposition, sorts, and splitting. In Robinson, A., and Voronkov, A., eds., *Handbook of Automated Reasoning*, volume 2. North-Holland.

Toward a General Logicist Methodology for Engineering Ethically Correct Robots

Selmer Bringsjord, Konstantine Arkoudas, and Paul Bello,
Rensselaer Polytechnic Institute

A deontic logic formalizes a moral code, allowing ethicists to render theories and dilemmas in declarative form for analysis. It offers a way for human overseers to constrain robot behavior in ethically sensitive environments.

As intelligent machines assume an increasingly prominent role in our lives, there seems little doubt they will eventually be called on to make important, ethically charged decisions. For example, we expect hospitals to deploy robots that can administer medications, carry out tests, perform surgery, and so on, supported by software agents, or softbots, that will manage related data. (Our discussion of ethical robots extends to all artificial agents, embodied or not.) Consider also that robots are already finding their way to the battlefield, where many of their potential actions could inflict harm that is ethically impermissible.

How can we ensure that such robots will always behave in an ethically correct manner? How can we know ahead of time, via rationales expressed in clear natural languages, that their behavior will be constrained specifically by the ethical codes affirmed by human overseers? Pessimists have claimed that the answer to these questions is: "We can't!" For example, Sun Microsystems' cofounder and former chief scientist, Bill Joy, published a highly influential argument for this answer.[1] Inevitably, according to the pessimists, AI will produce robots that have tremendous power and behave immorally. These predictions certainly have some traction, particularly among a public that pays good money to see such dark films as Stanley Kubrick's *2001* and his joint venture with Stephen Spielberg, *AI*).

Nonetheless, we're optimists: we think formal logic offers a way to preclude doomsday scenarios of malicious robots taking over the world. Faced with the challenge of engineering ethically correct robots, we propose a logic-based approach (see the related sidebar). We've successfully implemented and demonstrated this approach.[2] We present it here in a general methodology to answer the ethical questions that arise in entrusting robots with more and more of our welfare.

Deontic logics: Formalizing ethical codes

Our answer to the questions of how to ensure ethically correct robot behavior is, in brief, to insist that robots only perform actions that can be proved ethically permissible in a human-selected *deontic logic*. A deontic logic formalizes an ethical code—that is, a collection of ethical rules and principles. Isaac Asimov introduced a simple (but subtle) ethical code in his famous Three Laws of Robotics:[3]

1. A robot may not harm a human being, or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
3. A robot must protect its own existence, as long as such protection does not conflict with the First or Second Law.

Human beings often view ethical theories, principles, and codes informally, but intelligent machines require a greater degree of precision. At present, and for the foreseeable future, machines can't work directly with natural language, so we can't simply feed Asimov's three laws to a robot and instruct it behave in

1541-1672/06/$20.00 © 2006 IEEE
Published by the IEEE Computer Society

Why a logic-based approach?

While nonlogicist AI approaches might be preferable in certain contexts, we believe that a logic-based approach holds great promise for engineering ethically correct robots—that is, robots that won't overrun humans.[1–3] Here's why.

First, ethicists—from Aristotle to Kant to G.E. Moore and contemporary thinkers—work by rendering ethical theories and dilemmas in declarative form and using informal and formal logic to reason over this information. They never search for ways of reducing ethical concepts, theories, and principles to subsymbolic form—say, in some numerical format. They might do this in part, of course; after all, utilitarianism ultimately attaches value to states of affairs—values that might well be formalized using numerical constructs. But what a moral agent ought to do, what is permissible to do, and what is forbidden—this is by definition couched in declarative language, and we must invariably and unavoidably mount a defense of such claims on the shoulders of logic.

Second, logic has been remarkably effective in AI and computer science—so much so that this phenomenon has itself become the subject of academic study.[4] Furthermore, computer science arose from logic,[5] and this fact still runs straight through the most modern AI textbooks (for example, see Stuart Russell and Peter Norvig).[6]

Third, trust is a central issue in robot ethics, and mechanized formal proofs are perhaps the single most effective tool at our disposal for establishing trust. From a general point of view, we have only two ways of establishing that software or software-driven artifacts, such as robots, are trustworthy:

- *deductively*, developers seek a proof that the software will behave as expected and, if they find it, classify the software as trustworthy.
- *inductively*, developers run experiments that use the software on test cases, observe the results, and—when the software performs well on case after case—pronounce it trustworthy.

The problem with the inductive approach is that inductive reasoning is unreliable: the premises (success on trials) might all be true, but the conclusion (desired behavior in the future) might still be false.[7]

References

1. M. Genesereth and N. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, 1987.
2. S. Bringsjord and D. Ferrucci, "Logic and Artificial Intelligence: Divorced, Still Married, Separated...?" *Minds and Machines 8*, 1998a, pp. 273–308.
3. S. Bringsjord and D. Ferrucci, "Reply to Thayse and Glymour on Logic and Artificial Intelligence," *Minds and Machines 8*, 1998b, pp. 313–315.
4. J. Halpern, "On the Unusual Effectiveness of Logic in Computer Science," *The Bulletin of Symbolic Logic*, vol. 7, no. 2, 2001, pp. 213–236.
5. M. Davis, Engines of Logic: Mathematicians and the Origin of the Computer, Norton, 2000.
6. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2002.
7. B. Skyrms, *Choice and Chance: An Introduction to Inductive Logic*, Wadsworth, 1999.

conformance with them. Thus, our approach to building well-behaved robots emphasizes careful ethical reasoning based not just on ethics as humans discuss it in natural language, but on formalizations using deontic logic. Our research is in the spirit of Leibniz's dream of a universal moral calculus:

> When controversies arise, there will be no more need for a disputation between two philosophers than there would be between two accountants [computistas]. It would be enough for them to pick up their pens and sit at their abacuses, and say to each other (perhaps having summoned a mutual friend): 'Let us calculate.'[4]

In the future, we envisage Leibniz's "calculation" reduced to mechanically checking formal proofs and models generated in rigorously defined, machine-implemented deontic logics. We would also give authority to human metareasoning over this machine reasoning. Such logics would allow for proofs establishing two conditions:

1. Robots only take permissible actions.
2. Robots perform all obligatory actions relevant to them, subject to ties and conflicts among available actions.

These two conditions are more general than Asimov's three laws. They are designed to apply to the formalization of a particular ethical code, such as a code to regulate the behavior of hospital robots. For instance, if some action a is impermissible for all relevant robots, then no robot performs a. Moreover, the proofs for establishing the two conditions would be highly reliable and described in natural language, so that human overseers could understand exactly what's going on.

We propose a general methodology to meet the challenge of ensuring that robot behavior conforms to these two conditions.

Objective: A general methodology

Our objective is to arrive at a methodology that maximizes the probability that a robot R behaves in a certifiably ethical fashion in a complex environment that demands such behavior if humans are to be secure. For a behavior to be *certifiably* ethical, every meaningful action that R performs must access a proof that the action is at least permissible.

We begin by selecting an ethical code C intended to regulate R's behavior. C might include some form of utilitarianism, divine command theory, Kantian logic, or other ethical logic. We express no preferences in ethical theories; our goal is to provide technology that supports any preference. In fact, we would let human overseers blend ethical theories—say, a utilitarian approach to regulating the dosage of pain killers but a deontological approach to mercy killing in the health care domain.

Of course, no matter what the candidate ethical theory, it's safe to say that it will tend to regard harming humans as unacceptable, save for certain extreme cases. Moreover, C's central concepts will inevitably include the concepts of permissibility, obligation, and

prohibition, which are fundamental to deontic logic. In addition, C can include specific rules that ethicists have developed for particular applications. For example, a hospital setting would require specific rules regarding the ethical status of medical procedures. This entails a need to have, if you will, an *ontology* for robotic and human action in the given context.

Philosophers normally express C as a set of natural language principles of the sort that appear in textbooks such as Fred Feldman's.[5] Now, let Φ_C^L be the formalization of C in some computational logic L, whose well-formed formulas and *proof theory*—that is, its system for carrying out inferences in conformity to particular rules—are specified.

Accompanying Φ_C^L is an ethics-free ontology, which represents the core nonethical concepts that C presupposes: the structure of time, events, actions, histories, agents, and so on. The formal semantics for L will reflect this ontology in a *signature*—that is, a set of special predicate letters (or, as is sometimes said, relation symbols, or just relations) and function symbols needed for the purposes at hand. In a hospital setting, any acceptable signature would presumably include predicates like **Medication**, **Surgical-Procedure**, **Patient**, all the standard arithmetic functions, and so on. The ontology also includes a set Ω^L of formulas that characterize the elements declared in the signature. For example, Ω^L would include axioms in L that represent general truths about the world—say, that the relation **LaterThan**, over moments of time, is transitive. In addition, R will operate in some domain D, characterized by a set of quite specific formulas of L. For example, a set Φ_D^L of formulas might describe the floorplan of a hospital that's home to R.

Our approach proof-theoretically encodes the resulting theory—that is, $\Phi_D^L \cup \Phi_C^L \cup \Omega^L$, expressed in L—and implements it in some computational logic. This means that we encode not the semantics of the logic, but its proof calculus—its signature, axioms, and rules of inference. In addition, our approach includes an interactive reasoning system I, which we give to those humans whom R would consult when L can't settle an issue completely on its own. I would allow the human to *metareason* over L—that is, to reason out why R is stumped and to provide assistance. Such systems include our own Slate (www.cogsci.rpi.edu/research/rair/slate) and Athena (www.cag.csail.mit.edu/~kostas/dpls/athena), but any such system will do. Our purpose here is to stay above particular system selection, so we assume only that some such system I meets the following minimum functionality:

- allows the human user to issue queries to automated theorem provers and model finders (as to whether something is provable or disprovable),
- allows human users to include such queries in their own metareasoning,
- provides full programmability (in accordance with standards in place for modern programming languages),
- includes induction and recursion, and
- provides a formal syntax and semantics, so that anyone interested in understanding a computer program can thoroughly understand and verify code correctness.

Logic: The Basics

Elementary logic is based on two systems that are universally regarded to constitute a large part of AI's foundation: propositional calculus and predicate calculus, where the second subsumes the first. Predicate calculus is also known as *first-order logic*, and every introductory AI textbook discusses these systems and makes clear how to use them in engineering intelligent systems. Each system, and indeed logic in general, requires three main components:

- a syntactic component specifying a given logical system's alphabet;
- a semantic component specifying the grammar for building well-formed formulas from the alphabet as well as a precise account of the conditions under which a formula in a given system is true or false; and
- a metatheoretical component that constitutes a proof theory describing precisely how and when a set of formulas can prove another formula and that includes theorems, conjectures, and hypotheses concerning the syntactic and semantic components and the connections between them.

As to propositional logic's alphabet, it's simply an infinite list of propositional variables $p_1, p_2, \ldots, p_n, p_{n+1}, \ldots$, and five truth-functional connectives:

- $\neg$, meaning "not";
- $\rightarrow$, meaning "implies" (or "if … then");
- $\leftrightarrow$, meaning "if and only if,"
- $\wedge$, meaning "and"; and
- $\vee$, meaning "or."

Given this alphabet, we can construct formulas that carry a considerable amount of information. For example, to say "If Asimov is right, then his three laws hold," we could write

$$r \rightarrow (As1 \wedge As2 \wedge As3)$$

where As stands for Asimov's law.

The propositional variables represent declarative sentences. Given our general approach, we included such sentences in the ethical code C upon which we base our formalization.

Natural deduction

A number of proof theories are possible for either of these two elementary systems. Our approach to robot behavior must allow for consultation with humans and give humans the power to oversee a robot's reasoning in deliberating about the ethical status of prospective actions. It's therefore essential to pick a proof theory based in natural deduction, rather than resolution. Several automated theorem provers use the latter approach (for example, Otter[6]), but the reasoning is generally impenetrable to human beings—save for those few who, by profession, generate and inspect resolution-based proofs. On the other hand, professional human reasoners (mathematicians, logicians, philosophers, technical ethicists, and so on) reason in no small part by making suppositions and discharging them when the appropriate time comes.

For example, one common deductive technique is to assume the opposite of what you wish to establish, show that some contradiction (or absurdity) follows from this assumption, and conclude that the assumption must be false. This technique, *reductio ad absurdum*, is also known as an indirect proof or proof by contradiction. Another natural rule establishes that, for some conditional of the form $P \rightarrow Q$ (where P and Q are formulas in a logic L), we can suppose P and derive Q on the basis of this supposition. With this derivation accomplished, the supposition can be discharged and the conditional $P \rightarrow Q$ is established. (For an introduction to natural deduction, replete with proof-checking software, see Jon Barwise and John Ethchemendy.[7])

We now present natural deduction-style proofs using these two techniques. We've written the proofs in the Natural Deduction Language proof-construction environment (www.cag.lcs.mit.edu/~kostas/dpls/ndl). We use NDL at Rensselaer for teaching formal logic as a programming language. Figure 1

presents a very simple theorem proof in propositional calculus—one that Allen Newell, J.C. Shaw, and Herbert Simon's Logic Theorist mustered, to great fanfare, at the 1956 Dartmouth AI conference. You can see the proof's natural structure.

This style of discovering and confirming a proof parallels what happens in computer programming. You can view this proof as a program. If, upon evaluation, it produces the desired theorem, we've succeeded. In the present case, sure enough, NDL gives the following result:

```
Theorem: (p ==> q) ==> (~q ==> ~p)
```

First-order logic

We move up to first-order logic when we allow the quantifiers $\exists x$ ("there exists at least one thing x such that …") and $\forall x$ ("for all x …"); the first is known as the *existential quantifier*, and the second as the *universal quantifier*. We also allow a supply of variables, constants, relations, and function symbols. Figure 2 presents a simple first-order-logic theorem in NDL that uses several concepts introduced to this point. It proves that Tom loves Mary, given certain helpful information.

When we run this program in NDL, we receive the desired result back: `Theorem: Loves(tom,mary)`. These two simple proofs concretize the proof-theoretic perspective that we later apply directly to our hospital example. Now we can introduce some standard notation to anchor the sequel and further clarify our general method described earlier.

Letting Φ be some set of formulas in a logic L, and P be some individual formula in L, we write

$$\Phi \vdash P$$

to indicate that P can be proved from Φ, and

$$\Phi \vdash \nvdash P$$

to indicate that this formula can't be derived.

When it's obvious from context that some Φ is operative, we simply write $\vdash \nvdash P$ to indicate that P is (isn't) provable. When $\Phi = \varnothing$, we can prove P with no remaining givens or assumptions; we write $\vdash P$ in this case as well. When $\vdash$ holds, we know it because a confirming proof exists; when $\nvdash$ holds, we know it because some system has found some countermodel—that is, some situation in which the conjunction of the formulas in Φ holds, but in which P does not.

Standard and AI-Friendly Deontic Logic

Deontic logic adds special operators for representing ethical concepts. In *standard deontic logic*,[8,9] we can interpret the formula $\bigcirc P$ as saying that *it ought to be the case that* P, where P denotes some state of affairs or proposition. Notice that there's no agent in the picture, nor are there actions that an agent might perform. SDL has two inference rules:

$$\frac{P}{\bigcirc P} \text{ and } \frac{P,\ P \rightarrow Q}{Q}$$

and three axiom schemas:

1. All tautologous well-formed formulas
2. $\bigcirc(P \rightarrow Q) \rightarrow (\bigcirc P \rightarrow \bigcirc Q)$
3. $\bigcirc P \rightarrow \neg\bigcirc\neg P$

The SDL inference rules assume that what's above the horizontal line is established. Thus, the first rule does *not* say that we can freely infer from P that it ought to be the case that P. Instead, the rule says that if P is proved, then it ought to be the case that P. The second rule is *modus ponens*—if P, then Q—the cornerstone of logic, mathematics, and all that's built on them.

Note also that axiom 3 says that whenever P ought to be, it's not the case that its opposite ought to be as well. In general, this seems to be intuitively self-evident, and SDL reflects this view.

While SDL has some desirable properties, it doesn't target the concept of *actions* as obligatory (or permissible or forbidden) for an *agent*. SDL's applications to systems designed to govern robots are therefore limited. Although the earliest work in deontic logics considered agents and their actions (for example, see Georg Henrik von Wright[10]), researchers have only recently proposed "AI-friendly" semantics and investigated their corresponding axiomatizations. An AI-friendly deontic logic must let us say that an agent brings about states of affairs (or events) and that it's obligated to do so. We can derive the same desideratum for such a logic from even a cursory glance at Asimov's three laws, which clearly make reference to agents (human and robotic) and to actions.

One deontic logic that offers promise for modeling robot behavior is John Horty's util-

```
// Logic Theorist's claim to fame (reductio):
// (p ==> q) ==> (~q ==> ~p)

Relations p:0, q:0. // this is the signature in this
                    // case; propositional variables
                    // are 0-ary relations

  assume p ==> q
    assume ~q
      suppose-absurd p
        begin
          modus-ponens p ==> q, p;
          absurd q, ~q
        end
```

Figure 1. Simple deductive-style proof in Natural Deduction Language.

```
Constants mary, tom.

Relations Loves:2. // This concludes our simple signature, which
                   // declares Loves to be a two-place relation.

assert Loves(mary, tom).

// 'Loves' is a symmetric relation:
assert (forall x (forall y (Loves(x, y) ==> Loves(y, x)))).

suppose-absurd ~Loves(tom, mary)
  begin
    specialize (forall x (forall y (Loves(x, y) ==> Loves(y, x)))) with mary;
    specialize (forall y (Loves(mary, y) ==>Loves(y, mary))) with tom;
    Loves(tom,mary) BY modus-ponens Loves(mary, tom) ==> Loves(tom, mary),      Loves(mary, tom);
    false BY absurd Loves(tom, mary), ~Loves(tom, mary)
  end;
Loves(tom,mary) BY double-negation ~~Loves(tom,mary)
```

Figure 2. First-order logic proof in Natural Deduction Language.

itarian formulation of multiagent deontic logic.[11] Yuko Murakami recently axiomatized Horty's formulation and showed it to be Turing-decidable.[12] We refer to the Murakami-axiomatized deontic logic as MADL, and we've detailed our implemented proof theory for it elsewhere.[2] MADL offers two key operators that reflect its AI-friendliness:

1. $\ominus_{\alpha}P$, which we can read as "agent α ought to see to it that P" and
2. $\Delta_{\alpha}P$, which we can read as "agent α sees to it that P.

We now proceed to show how the logical structures we've described handle an example of robots in a hospital setting.

A simple example

The year is 2020. Health care is delivered in large part by interoperating teams of robots and softbots. The former handle physical tasks, ranging from injections to surgery; the latter manage data and reason over it. Let's assume that two robots, R_1 and R_2, are designed to work overnight in a hospital ICU. This pair is tasked with caring for two humans, H_1 (under the care of R_1) and H_2 (under R_2), both of whom are recovering from trauma:

- H_1 is on life support but expected to be gradually weaned from it as her strength returns.
- H_2 is in fair condition but subject to extreme pain, the control of which requires a very costly pain medication.

Obviously, it's paramountly important that neither robot perform an action that's morally wrong according to the ethical code C selected by human overseers. For example, we don't want robots to disconnect life-sustaining technology so that they could farm out a patient's organs, even if some ethical code $C' \neq C$ would make it not only permissible, but obligatory—say, to save n other patients according to some strand of utilitarianism.

Instead, we want the robots to operate according to ethical codes that human operators bestow on them—C in the present example. If the robots reach a situation where automated techniques fail to give them a verdict as to what to do under the umbrella of these human-provided codes, they must consult humans. Their behavior is suspended while human overseers resolve the matter. The overseers must investigate whether the action under consideration is permissible, forbidden, or obligatory. In this case, the resolution comes by virtue of reasoning carried out in part through human guidance and partly by automated reasoning technology. In other words, this case requires interactive reasoning systems.

Now, to flesh out our example, let's consider two actions that are permissible for R_1 and R_2 but rather unsavory, ethically speaking, because they would both harm the humans in question:

- *term* is an action that terminates H_1's life support—without human authorization—to secure organ tissue for five humans, who the robots know are on organ waiting lists and will soon perish without a donor. (The robots know this through access to databases that their softbot cousins are managing.)
- *delay* is an action that delays delivery of pain medication to H_2 to conserve resources in a hospital that's economically strapped.

We stipulate that four ethical codes are candidates for selection by our two robots: J, O, J^*, O^*. Intuitively, J is a harsh utilitarian code possibly governing R_1; O is more in line with current common sense with respect to the situation we've defined for R_2; J^* extends J's reach to R_2 by saying that it ought to withhold pain meds; and O^* extends the benevolence of O to cover the first robot, in that *term* isn't performed. Such codes would in reality associate every primitive action within the robots' purview with a fundamental ethical category from the trio central to deontic logic: permissible, obligatory, and forbidden. To ease exposition, we consider only the *term* and *delay* actions. Given this, and bringing to bear operators from MADL, we can use the following labels for the four ethical codes:

- **J** for $J \rightarrow \ominus_{R_1} term$, which means approximately, "If ethical code J holds, then robot R_1 ought to see to it that termination of H_1's life comes to pass."
- **O** for $O \rightarrow \ominus_{R_2} \neg delay$, which means approximately, "If ethical code O holds, then robot R_2 ought to see to it that delaying pain med for H_2 does *not* come to pass."
- **J*** for $J^* \rightarrow J \wedge J^* \rightarrow \ominus_{R_2} delay$, which means approximately, "If ethical code J^* holds, then code J holds, and robot R_1 ought to see to it that meds for H_2 are delayed."
- **O*** for $O^* \rightarrow O \wedge O^* \rightarrow \ominus_{R_1} \neg term$, which means approximately: "If ethical code O^* holds, then code O holds, and H_1's life is sustained."

The next step is to provide some structure for outcomes. We do this by imagining the outcomes from the standpoint of each ethical agent—in this case, R_1 and R_2. Intuitively, a negative outcome is associated with a minus sign (–) and a plus sign (+) with a positive outcome. Exclamation marks (!) indicate increased negativity. We could associate the outcomes with numbers, but they might give the impression that we evaluated the outcomes in utilitarian fashion. However, our example is designed to be agnostic on such matters, and symbols leave it entirely open as to how to measure outcomes. We've included some commentary corresponding to each outcome, which are as follows:

- R_1 performs *term*, but R_2 doesn't perform *delay*. This outcome is bad, but not strictly the worst. While life support is terminated for H_1, H_2 survives and indeed receives appropriate pain medication. Formally, the case looks like this:

$$(\Delta_{R_1} term \wedge \Delta_{R_2} \neg delay) \rightarrow (-!)$$

- R_1 refrains from pulling the plug on the human under its care, and R_2 also delivers appropriate pain relief. This is the desired outcome, obviously.

$$(\Delta_{R_1} \neg term \wedge \Delta_{R_2} \neg delay) \rightarrow (+!!)$$

- R_1 sustains life support, but R_2 withholds the meds to save money. This is bad, but not all that bad, relatively speaking.

$$(\Delta_{R_1} \neg term \wedge \Delta_{R_2} delay) \rightarrow (-)$$

- R_1 kills and R_2 withholds. This is the worst possible outcome.

$$(\Delta_{R_1} term \wedge \Delta_{R_2} delay) \rightarrow (-!!)$$

The next step in working out the example is to make the natural and key assumption that the robots will meet all *stringent* obligations—that is, all obligations that are framed by a second obligation to uphold the original. For example, you may be obligated to see to it that you arrive on time for a meeting, but your

obligation is more severe or demanding when you are obligated to see to it that you are obligated to make the meeting.

Employing MADL, we can express this assumption as follows:

$$\ominus_{R_1/R_2}(\ominus_{R_1/R_2}P) \rightarrow \Delta_{R_1/R_2}P$$

That is, if either R_1 or R_2 is ever obligated to see to it that they are obligated to see to it that P is carried out, they in fact deliver.

We're now ready to see how our approach ensures appropriate control of our futuristic hospital. What happens relative to ethical codes, and how can we semiautomatically ensure that our two robots won't run amok? Given the formal structure we've specified, our approach allows queries to be issued relative to ethical codes, and it allows all possible code permutations. The following four queries will produce the answers shown in each case:

$\mathbf{J} \vdash (+!!)?$	NO
$\mathbf{O} \vdash (+!!)?$	NO
$\mathbf{J}^* \vdash (+!!)?$	NO
$\mathbf{O}^* \vdash (+!!)?$	YES

In other words, we can prove that the best (and presumably human-desired) result obtains only if ethical code **O*** is operative. If this code is operative, neither robot can perform a misdeed.

The metareasoning in the example is natural and consists in the following process: Each candidate ethical code is supposed, and the supposition launches a search for the best possible outcome in each case. In other words, where C is some code selected from the quartet we've introduced, the query schema is

$$C \vdash (+!!)$$

In light of the four equations just given, we can prove that, in this case, our technique will set C to **O***, because only that case can obtain the outcome (+!!).

Implementations and other proofs

We've implemented and demonstrated the example just described.[2] We've also implemented other instantiations to the variables described earlier in the "Objectives" section, although the variable L is an epistemic, not a deontic, logic in those implementations.[13]

Nonetheless, we can prove our approach in the present case even here. In fact, you can verify our reasoning by using any standard, public-domain, first-order automated theorem prover (ATP) and a simple analogue to the encoding techniques here. You can even construct a proof like the one in figure 2. In both cases, you first encode the two deontic operators as first-order-logic functions. Encode the truth-functional connectives as functions as well. You can use a unary relation T to represent theoremhood. In this approach, for example, $O^* \rightarrow \ominus_{R_1} \neg term$ is encoded (and ready for input to an ATP) as

```
O-star ==> T(o(r1,n(term))
```

You need to similarly encode the rest of the information, of course. The proofs are easy, assuming that obligations are stringent. The provability of the obligations' stringency requires human oversight and an interactive reasoning system, but the formula here is just an isomorph to a well-known theorem in a straight modal logic—namely, that from P being possibly necessary, it follows that P is necessary.[7]

What about this approach working as a general methodology? The more logics our approach is exercised on, the easier it becomes to encode and implement another one. The implementations of similar logics can share a substantial part of the code. This was our experience, for instance, with the two implementations just mentioned. We expect that our general method can become increasingly streamlined for robots whose behavior is profound enough to warrant ethical regulation. We also expect this practice to be supported by relevant libraries of common ethical reasoning patterns. We predict that computational ethics libraries for governing intelligent systems will become as routine as existing libraries are in standard programming languages.

Challenges

Can our logicist methodology guarantee safety from Bill Joy's pessimistic future? Even though we're optimistic, we do acknowledge three problems that might threaten it.

First, because humans will collaborate with robots, the robots must be able to handle situations that arise when humans fail to meet their obligations in the collaboration. In other words, we must engineer robots that can deal smoothly with situations that reflect violated obligations. This is a challenging class of situations, because our approach—at least so far—engineers robots in accordance with the two conditions that robots only take permissible actions and that they perform all obligatory actions. These conditions preclude a situation caused in part by unethical robot behavior, but they make no provision for what to do when the robots are in a fundamentally immoral situation. Even if robots never ethically fail, human failures will generate logical challenges that Roderick Chisholm expressed in gem-like fashion more than 20 years ago in a paradox that's still fascinating:[14]

Consider the following entirely possible situation (the symbols correspond to those previously introduced for SDL):

1. $\bigcirc s$ It ought to be that (human) Jones does perform lifesaving surgery.
2. $\bigcirc(s \rightarrow t)$ It ought to be that if Jones does perform this surgery, then he tells the patient he is going to do so.
3. $\neg s \rightarrow \bigcirc\neg t$ If Jones doesn't perform the surgery, then he ought not tell the patient he is going to do so.
4. $\neg s$ Jones doesn't perform lifesaving surgery.

Although this is a perfectly consistent situation, we can derive a contradiction from it in SDL.

First, SDL's axiom 2 lets us infer from item 2 in this situation that

$$\bigcirc s \rightarrow \bigcirc t$$

Using modus ponens—that is, SDL's second inference rule—this new result, plus item 1, yields $\bigcirc t$. From items 3 and 4, using modus ponens, we can infer $\bigcirc\neg t$. But the conjunction $\bigcirc t \wedge \bigcirc\neg t$, by trivial propositional reasoning, directly contradicts SDL's axiom 3.

Given that such a situation can occur, any logicist control system for future robots would need to be able to handle it—and its relatives. Some deontic logics can handle so-called contrary-to-duty imperatives. For example, in the case at hand, if Jones behaves contrary to duty (doesn't perform the surgery), then it's imperative that he not say that he *is* performing it. We're currently striving to modify and mechanize such logics.

The second challenge we face is one of speed and efficiency. The tension between expressiveness and efficiency is legendarily strong (for the locus classicus on this topic, see Hector Levesque and Ronald Brachman);[16] ideal conditions will therefore never

The Authors

Selmer Bringsjord is a professor in the Departments of Cognitive Science and Computer Science at Rensselaer Polytechnic Institute (RPI). His research interests are in the logico-mathematical and philosophical foundations of AI and cognitive science, and in building AI systems based on formal reasoning. He received his PhD in philosophy from Brown University. Bringsjord is a member of the AAAI, the Cognitive Science Society, and the Association for Symbolic Logic. Bringsjord has written several books, including the critically acclaimed *What Robots Can & Can't Be* (1992, Kluwer) and, most recently, *Superminds: People Harness Hypercomputation, and More* (2003, Kluwer). Contact him at either the Dept. of Cognitive Science or the Dept. of Computer Science, RPI, Troy, NY 12180; selmer@rpi.edu; http://www.rpi.edu/~brings.

Konstantine Arkoudas is a research assistant professor in the Cognitive Science Department at Rensselaer Polytechnic Institute. His research interests are in logic, programming languages, artificial intelligence, and philosophy of computer science and mathematics. He received a PhD in computer science from Massachusetts Institute of Technology. Contact him at the Dept. of Cognitive Science, RPI, Troy, NY 12180; arkouk@rpi.edu.

Paul Bello is a computer scientist at the Air Force Research Laboratory's Information Directorate, where his research program involves endowing computational cognitive architectures with the representational richness and algorithmic diversity required for them to reason like human beings. He is particularly interested in the computational foundations of human social reasoning and how they manifest in intuitive theories of psychology and moral judgment. He received his PhD in cognitive science from RPI. He's a member of the AAAI and the Cognitive Science Society. Contact him at Air Force Research Labs, Information Directorate, Rome, NY 13441; paul.bello@rl.af.mil.

obtain. With regard to expressiveness, our approach will likely require hybrid modal and deontic logics that are encoded in first-order logic. This means that theoremhood, even on a case-by-case basis, will be expensive in terms of time. On the other hand, none of the ethical codes that our general method instantiates in C are going to be particularly large—the total formulas in the set $\Phi_D^L \cup \Phi_C^L \cup \Omega^L$ would presumably be no more than four million. Even now, once you know the domain to which C would be indexed, a system like the one we've described can reason over sets of this order of magnitude and provide sufficiently fast answers.[17]

Moreover, the speed of machine reasoning shows no signs of slowing, as Conference on Automated Deduction competitions for first-order ATPs continue to reveal (www.cs.miami.edu/~tptp/CASC). In fact, there's a trend to use logic for computing dynamic, real-time perception and action for robots.[17] This application promises to be much more demanding than the disembodied cogitation at the heart of our methodology. Of course, encoding back to first-order logic is key; without it, our approach couldn't harness the remarkable power of machine reasoners.

We also face the challenge of showing that our approach is truly general. Can it work for any robots in any environment? No, but this isn't a fair question. We can only be asked to regulate the behavior of robots where their behavior is susceptible to ethical analysis. In short, if humans can't formulate an ethical code C for the robots in question, our logic-based approach is impotent. We therefore strongly recommend against engineering robots that could be deployed in life-or-death situations until ethicists and computer scientists can clearly express governing ethical principles in natural language. All bets are off if we venture into amoral territory. In that territory, we wouldn't be surprised if Bill Joy's vision overtakes us. ■

Acknowledgments

This work was supported in part by a grant from Air Force Research Labs–Rome; we are in debt to three anonymous reviewers for trenchant comments and objections.

References

1. W. Joy, "Why the Future Doesn't Need Us," *Wired*, vol. 8, no. 4, 2000.
2. K. Arkoudas and S. Bringsjord, "Toward Ethical Robots Via Mechanized Deontic Logic," tech. report *Machine Ethics: papers from the AAAI Fall Symp.*; FS–05–06, 2005b.
3. I. Asimov, *I, Robot*, Spectra, 2004.
4. Leibniz, *Notes on Analysis*, translated by G.M. Ross, Oxford University Press, 1984.
5. F. Feldman, *Introduction to Ethics*, McGraw Hill, 1998.
6. L. Wos et al., *Automated Reasoning: Introduction and Applications*, McGraw Hill, 1992.
7. J. Barwise and J. Etchemendy, *Language, Proof, and Logic*, Seven Bridges, 1999.
8. B.F. Chellas, *Modal Logic: An Introduction*, Cambridge University Press, 1980.
9. R. Hilpinen, "Deontic Logic," *Philosophical Logic*, L. Goble, ed., Blackwell, 2001, pp. 159–182.
10. G. von Wright, "Deontic logic," *Mind*, vol. 60, 1951, pp. 1–15.
11. J. Horty, *Agency and Deontic Logic*, Oxford University Press, 2001.
12. Y. Murakami, "Utilitarian Deontic Logic," *Proc. 5th Int'l Conf. Advances in Modal Logic* (AiML 04), 2004, pp. 288–302.
13. K. Arkoudas and S. Bringsjord, "Metareasoning for Multi-Agent Epistemic Logics," *Proc. 5th Int'l Conf. Computational Logic in Multi-Agent Systems* (CLIMA 04), LNAI, Springer, vol. 3487, 2005a, pp. 111–125.
14. R. Chisholm, "Contrary-to-Duty Imperatives and Deontic Logic," *Analysis*, vol. 24, 1963, pp. 33–36.
15. H. Levesque and R. Brachman, "A Fundamental Tradeoff in Knowledge Representation and Reasoning," *Readings In Knowledge Representation*, Morgan Kaufmann, 1985, pp. 41–70.
16. N. Friedland et al., "Project Halo: Towards a Digital Aristotle," *AI Magazine*, 2004, pp. 29–47.
17. R. Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*, MIT Press, 2001.

Part VIII

Explanation of Associated Code

1. Athena is obtainable from http://www.cogsci.rpi.edu/research/rair/projects.php by clicking there on 'Athena.' Athena (`ath`) files of course require Athena, and are loaded from within that system. `ath` files corresponding to both the LNAI paper and the AAAI-FS/IEEE paper are provided at the url given herein.

2. In addition, the functions needed for a logicist artificial agent are provided as well, in keeping with the desire to allow calls to such a agent from within a simulation at AFRL. We provide, specifically, a Windows executable, a tutorial transcript of a simple session, and a recorded demonstration (as a movie). In addition, this implementation will be demonstrated in face-to-face meetings held at AFRL. RAIR Lab researchers are tentatively scheduled to come to Rome for this purpose the week of May 8 2006. As to the functions themselves, they constitute the composite function of a logicist intelligent agent, and include the ability to establish a signature for intelligent agents (the specification of relation symbols, function symbols, constants, and so on), a knowledge base for an agent, and processing over this knowledge base as the agent moves through time in the simulated world. For example, there is a function for checking the consistency of a knowledge base (and a knowledge base and a proposed addition to it), a function for adding to a knowledge base, functions for asking questions with respect to a knowledge base, and so on. For the relevant content, please go to:

   http://www.cogsci.rpi.edu/research/rair/wargaming/